공학석사 학위논문

병렬 기계 스케줄링의 다중 목적함수를 고려한 시뮬레이션 기반 강화학습

– 조선소 형강 제조 공정 스케줄링 적용 –

Simulation-based Reinforcement Learning with Multi-Objective for Parallel Machine Scheduling Problems

– Application to Profile Shop in Shipbuilding Industry –

2023년 8월

서울대학교 대학원

조선해양공학과

남 소 현

병렬 기계 스케줄링의 다중 목적함수를 고려한 시뮬레이션 기반 강화학습

– 조선소 형강 제조 공정 스케줄링 적용 –

지도 교수 우 종 훈

이 논문을 공학석사 학위논문으로 제출함
2023년 8월

서울대학교 대학원
조선해양공학과
남 소 현

남소현의 공학석사 학위논문을 인준함
2023년 8월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

초 록

조선소 형강공정은 선박의 블록 제작에 필요한 보강재를 생산하는 공정으로, 최근 조선소 수주 호황에 따른 생산 물량의 증가와 함께 생산성 향상이 요구되고 있다. 생산성 향상을 위한 방안으로는 생산 계획 최적화, 아웃소싱 물량 증대, 신규 설비 투입 등을 고려할 수 있다. 하지만 신규 설비 투입은 초기 투자 비용이 크고 향후 조선 산업의 사이클을 고려하면 근시안적인 해법에 불과하다. 또한 현재 조선소는 자체 생산 능력의 한계로 이미 상당한 물량에 대한 작업이 아웃소싱을 통해 이루어지고 있기 때문에 비용 측면에서 외주 물량을 증가시키는 것은 현실적으로 한계가 있다. 따라서 비용과 지속가능성을 고려했을 때 생산계획 최적화를 통한 생산성 향상이 대안이 될 수 있다.

조선 생산분야에서는 휴리스틱, 메타 휴리스틱, Integer Linear Programming 등의 방법론을 사용하여 생산 계획을 최적화하기 위한 많은 연구가 진행되어 왔다. 하지만 형강공정의 생산 유형인 병렬 기계 스케줄링 문제에 대한 연구는 거의 수행되지 않았다. 다른 제조분야에서는 병렬 기계 스케줄링에 대한 연구가 수행되어 왔지만, 실제 생산 현장에서는 변동성이 존재하거나 여러 지표를 동시에 달성하기 위한 의사결정이 진행됨에도, 변동성을 고려하지 않거나 단일한 목적함수만을 달성하기 위한 연구만 수행되었다. 따라서 본 연구에서는 조선소에 존재하는 병렬 기계 스케줄링 문제를 실제 생산 환경의 변동성을 반영하고, 실제 현장 관리자가 고려할 수 있는 여러 목적함수를 효과적으로 달성하기 위한 스케줄링 최적화 연구를 수행하였다.

구체적으로 조선소 형강공장의 생산성 향상을 위하여 작업의 투입과 작업 시간의 변동성이 존재하는 환경에 있어 납기 지연 최소화와

셋업 변경 회수 최소화를 목적함수로 하는 동적 스케줄링 알고리즘을 제안한다. 형강공정의 작업 순서 결정 문제에 대하여 두 목적함수가 고려된 Markov Decision Process 모델을 정의하고, 정책 기반 강화학습인 Proximal Policy Optimization 알고리즘을 사용하여 최적의 스케줄링 정책을 학습하였다. 다음으로 실적 데이터를 샘플링하여 구성한 테스트 시나리오들에 대해 우선순위규칙 (SSPT, ATCS, MDD, COVERT rule)과의 비교를 통해 개발된 알고리즘의 성능을 평가하였다. 결과적으로 본 연구에서 제안한 알고리즘은 평균 셋업 횟수 및 평균 납기지연의 두 가지 지표를 종합적으로 고려했을 때 우선순위 규칙들을 능가함을 확인하였다.

마지막으로, 본 연구에서 제안한 스케줄링 알고리즘을 조선소의 다른 병렬 기계 문제에도 적용 가능한지 검토하기 위해 분포함수로 구성된 일반적인 병렬 기계 스케줄링 문제를 가정하여 학습 및 우선순위규칙과의 성능 비교를 수행하였다. 결과적으로 본 연구에서 제안한 스케줄링 알고리즘이 다른 우선순위 규칙과 비교했을 때, 셋업 시간 최소화 및 납기지연 최소화라는 두 가지 목적함수를 효과적으로 달성함을 확인하였다.

주요어 : 강화학습, 병렬 기계 스케줄링, 이산 사건 시뮬레이션, 동적 스케줄링
학 번 : 2021-25405

목 차

표 목차

그림 목차

Abbreviation

PMSP	*Parallel Machine Scheduling Problem*
RL	*Reinforcement Learning*
MDP	*Markov Decision Process*
DQN	*Deep Q-Networks*
PPO	*Proximal Policy Optimization*
DES	*Discrete Event Simulation*
COTS	*Commercial Off-The-Shelf*
IAT	*Inter Arrival Time*
SSPT	*Shortest Processing Time and Shortest Setup Time*
ATCS	*Apparent Tardiness Cost with Setup*
MDD	*Modified Due Date*
COVERT	*Cost OVER Time*

Nomenclature

Chapter 2. Methodology

s, s'	*states*
a	*an action*
r	*a reward*
p	*state transition probability*
t	*discrete time step*
$S_t, \mathcal{S}$	*state at time t, set of all states*
$A_t, \mathcal{A}(s)$	*action at time t, set of all actions available in state s*
R_t	*reward at time t*
G_t	*return following time at t*
π	*policy (decision-making rule)*
π_*	*optimal policy*
γ	*discount rate*
$V_\pi(s)$	*value of state s under policy π*
$V_*(s)$	*value of state s under the optimal policy π_**
$Q_\pi(s, a)$	*value of taking action a in state s under policy π*
$Q_*(s, a)$	*value of taking action a in state s under the optimal policy π_**
α	*learning rate*
D	*replay memory in DQN*

Chapter 3. Problem Definition, Chapter 4 Modeling

i, j	*block or steel index*
k	*machine index*
B, B_i	*set of blocks, block i*
N	*the number of blocks*
M_k	*machine k*
m	*the number of machines*
n_i	*the number of T-bars (or steels) in block B_i*
$1/\lambda$	*average inter arrival time*
r_i	*release date of block B_i*
τ	*due date tightness*
d_i	*due date of block B_i*
S_i	*set of T-bars (or steels) included block B_i*
S_{ij}	*j^{th} T-bar (or steel) of block B_i*
n_{ij}	*the number of identical T-bar (or steel) S_{ij}*
t_{ij}	*welding thickness of T-bar (or steel) S_{ij}*
l_{ij}	*length of T-bar (or steel) S_{ij}*
θ_{ij}	*property of T-bar (or steel) S_{ij}*
θ_k	*set-up status of machine M_k*
v_{ij}	*working speed of steel S_{ij}*

$\overline{p}_{ij}$	*average processing time of steel* S_{ij}
δ_{pt}	*variability factor of processing time*
p_{ij}	*actual processing time of section steel* S_{ij}
C_i	*completion time of block* B_i
C_{max}	*makespan; maximum* C_i
T_i	*tardiness of block* B_i
σ_{ijk}	*set-up time when section steel* S_{ij} *is working on machine* M_k

Chapter 4 Modeling

t	*discrete time step or current time*
$Q(t)$	*queue of available steels at time* t
N_k	*the number of steels that satisfy condition* $\theta_k = \theta_{ij}$
N_q	*the number of steels in* $Q(t)$
r_{ij}	*the remaining processing time of steel* S_{ij}
$N_{r,i}$	*the number of remained steels in block* B_i
$t_{s,ij,k}$	*the start time of Steel* S_{ij} *in machine* M_k
k_1, k_2	*parameters of ATCS rule*
k	*parameter of COVRT rule*
$r_T, r_{T,i}$	*reward on tardiness, reward on tardiness of block* B_i
r_S	*reward on set-up time*
r_t	*reward at time* t

Chapter 6. Experiment

$\overline{T}$	*average tardiness of each block*
R_σ	*ratio of set-up of steels*

Chapter 7 Expansion to General Problem

i	*job index*
k	*machine index*
J_i	i^{th} *job*
p_i	*average processing time of job* J_i
d_i	*due date of job* J_i
n	*the number of jobs*
θ_i	*property of job* J_i
θ_k	*set-up status of machine* M_k
σ_{ik}	*set-up time when job* J_i *is working on machine* M_k
$\overline{T}$	*average tardiness of each job*
$\overline{\sigma}$	*average set-up time of each job*

제 1 장 서론

1.1 연구 동기 및 배경

최근 조선소에서는 수주 호황에 따른 조선소 생산 야드 내 작업 물량 증가에 대처하기 위하여 생산시스템의 생산성을 향상시키는 것이 중요한 문제가 되고 있다. 조선소의 야드에서 이루어지는 생산 시스템의 생산성 향상을 위한 방안으로는 생산 계획의 최적화, 새로운 설비의 투입, 추가적인 외주 작업의 실시 등을 고려할 수 있다. 이 중 새로운 설비의 투입과 외주 물량의 추가는 당장의 물량을 해소하는 데에는 도움이 되지만, 초기 투자 비용이 크다는 단점이 있다. 또한 향후 조선 산업의 사이클에 따라 하락기로 들어섰을 때를 고려하면 매우 근시안적인 해법에 해당한다. 반면, 생산 계획의 최적화는 생산 시스템 내의 불필요한 대기, 재고, 운반 등의 낭비 요소를 최소화함으로써 적은 비용으로 효율적인 생산성 개선이 가능하다. 현재 제조업 전반에 걸쳐 생산 계획 최적화에 대한 연구가 진행되고 있으며, 특히 조선 산업 분야에서도 다양한 연구가 진행되고 있다.

Lee and Kim (1995)은 Genetic algorithm을 이용한 탑재공장 부하 평준화에 대한 연구를 진행하였다. Lee and Kim (2011)은 한국의 대형 조선소를 대상으로 기수립된 일정 계획의 부하 평준화를 위한 휴리스틱 알고리즘을 개발하였다. Roh and Cha (2011)은 조선소 내에 존재하는 여러 개의 트랜스포터를 이용한 블록 운송에 대하여 메타 휴리스틱 알고리즘인 Ant colony algorithm과 Genetic algorithm을 이용하여 총 이동거리의 최소화를 목적으로 하는 최적화를 수행하였다. 손정열 et al.

(2014)은 자체적으로 제안한 블록의 적시 공급과 이송장비의 효율적인 운영을 위하여 블록 적치장 운영 계획 문제를 휴리스틱 알고리즘을 이용하여 해결하고자 하였다. Park and Kim (2020)은 Integer Linear Programming을 이용하여 의장공장의 물량 할당 최적화에 대한 연구를 수행하였다.

위와 같이 조선소의 생산 계획 최적화에 대한 연구가 다양한 방법론을 통하여 다수 진행되고 있음에도 불구하고, 제조업에 존재하는 다양한 생산 유형 중 하나인 병렬 기계 스케줄링에 대한 연구가 전무했다. 병렬 기계 스케줄링 문제(Parallel Machine Scheduling Problem; PMSP)는 단일 기계 스케줄링 문제의 일반적인 형태로 단일한 공정에서 병렬적으로 작업을 수행하는 기계들에 대한 작업 순서를 결정하는 문제이기 때문에 제조업의 스케줄링 문제를 모델링 하는 데에 많이 사용되고 있다(Pinedo 2012). 조선소에서 일어나는 선박 건조작업에 있어서도 병렬 기계 스케줄링 문제로 모델링 할 수 있는 생산 작업이 다수 존재한다. 생산 야드 전체 단위에서는 선박의 도크 할당 문제, 트랜스포터 할당 문제가 있고, 공장 단위에서는 형강공정에 존재하는 용접라인의 작업 투입 순서 문제 등이 병렬 기계 스케줄링 문제로 모델링 할 수 있다.

기존 연구에서 다루어진 스케줄링 문제들은 제조 라인의 특성에 따라 Pinedo (2012)에서 제안한 표기법으로 나타낼 수 있다. 구체적으로는 $\alpha|\beta|\gamma$ 의 triplet 구조로 표현하며, α 는 문제의 특성, β 는 문제의 제약, γ 는 목적함수를 의미한다. 본 연구에서 목적하는 병렬 기계는 그 특성에 따라 [그림 1-1]에 표현되어 있듯이 세 종류로 분류할 수 있다. 첫 번째는 가장 간단한 문제로 모든 병렬 기계의 특성(작업 속도 등)이 동일한 환경이며, P_m (Identical machines in parallel)으로 표현할 수 있다. P_m 은 동일한 m 개의 machine이

존재하는 문제로, 각 작업(job)이 어느 기계에서 작업해도 그 작업시간이 동일하다. 두 번째는 서로 다른 작업 속도를 가지는 m개의 machine에 대한 문제로, Q_m (Machines in parallel with different speeds)으로 표현할 수 있다. Q_m 의 경우, 각 작업 시간은 투입된 기계의 작업 속도에 의존한다. 마지막으로 각 작업과 m 개의 기계의 조합에 따라 작업시간이 달라지는 R_m (Unrelated machines in parallel)이 있다.

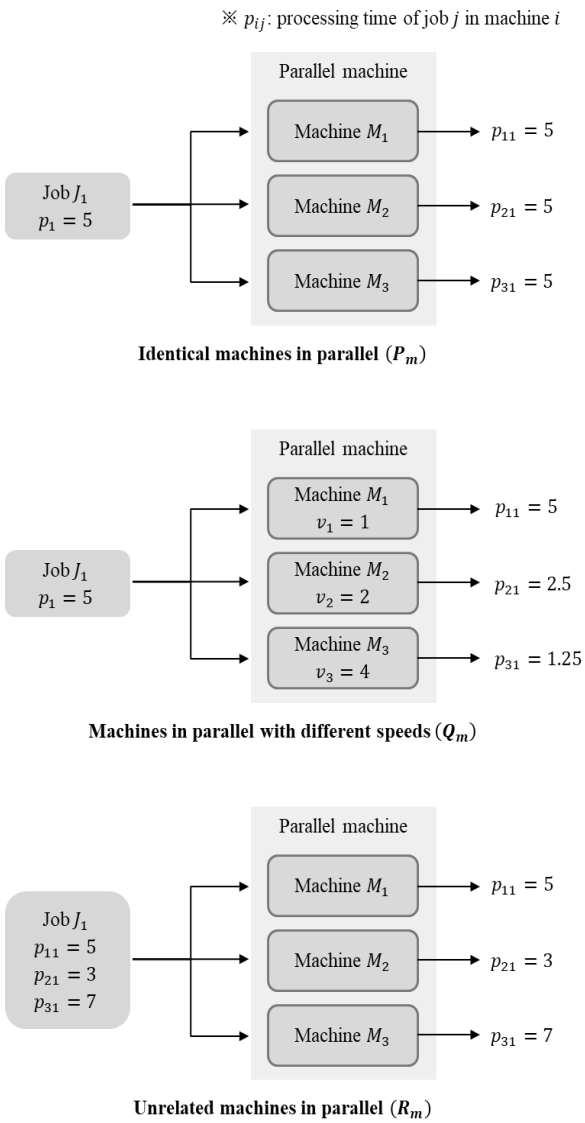


그림 1-1 Three types of Parallel Machine Problem

P_m, Q_m, R_m으로 모델링 할 수 있는 병렬 기계 스케줄링 문제는 제조 시스템뿐만 아니라 클라우드 컴퓨팅, 마이크로 컴퓨팅 등 다양한 분야에 응용이 가능하기 때문에 그동안 병렬 기계에서의 작업 할당을 최적화하려는 연구들이 많이 수행되어왔다. 대표적인 방법론으로는 휴리스틱 방법과 메타 휴리스틱 방법론이 있다. [표 1−1]에서는 병렬 기계 스케줄링 문제에 휴리스틱이나 메타 휴리스틱의 방법을 사용한 선행연구를 그 문제 특성과 함께 정리했다.

표 1-1 Related Works on PMSP using Heuristics and Meta-Heuristics

Author (year)	Problem Type	Algorithm
Kim et al. (2003)	$R_m \mid b, fmls, s_{fg} \mid \Sigma w_j T_j$	Simulated annealing
Chen and Chen (2009)	$R_m \mid s_{jk} \mid \Sigma w_j U_j$	Tabu search, Variable neighborhood descent
Lee et al. (2013)	$R_m \mid s_{ijk} \mid \Sigma T_j$	Tabu search
Chaudhry and Elbadawi (2017)	$P_m \mid\mid \Sigma T_j$	Genetic algorithm
Lee (2018)	$P_m \mid s_{ij} \mid \Sigma T_j$	ATCS_APD, Iterated greedy algorithm

Kim et al. (2003)은 순서 의존적 셋업 시간이 존재하는 병렬 기계 스케줄링 문제에 대하여 총 납기지연의 최소화를 목적함수로 lot을 고려하여 이웃해 탐색을 수행하는 Simulated annealing 기반 스케줄링 알고리즘을 제안하였다. Chen and Chen (2009)은 Variable neighborhood descent와 Tabu search를 결합한 병렬 기계 스케줄링

알고리즘을 개발하였고, 납기 지연 작업 수 최소화라는 목적함수를 고려하여 해의 탐색범위를 줄이기 위한 4가지의 이웃해 구조를 제안하였다. Lee et al. (2013)은 총 납기지연의 최소화를 목적함수로 순서 및 기계 의존적인 셋업 시간을 고려한 병렬 기계 스케줄링 문제를 다루었고, 스케줄링 알고리즘으로서 swap과 insertion 연산을 바탕으로 8가지 이웃해 탐색 전략을 정의하여 Tabu search를 적용하였다. Chaudhry and Elbadawi (2017)은 총 납기지연 최소화를 목적함수로 전체 작업의 처리 순서와 각 작업을 수행할 기계 정보로 chromosome을 구성하고 유전 알고리즘을 적용하여 스케줄링을 수행하였다. Lee (2018)은 병렬 기계 시스템으로 모델링 되는 Acrylonitrile-Butadiene-Styrene plate 제작 공정의 스케줄링 문제에 대하여 총 납기지연의 최소화를 위하여 두 단계로 구성된 스케줄링 알고리즘을 제안하였다. 해당 스케줄링 알고리즘은 먼저 ATCS_APD (the Apparent Tardiness Cost with Set-ups with the Adjacent Processing Time and Due date)라는 우선순위규칙을 통해 초기 계획을 수립하고, iterated greedy 기반의 탐색 방법을 적용하여 초기 계획을 개선했다.

[표 1-2]에서는 각 방법론의 장단점을 정리했다. 메타휴리스틱의 경우, 오랜 시간동안 해를 탐색하여 local optimum으로 수렴될 확률이 적어 좋은 성능의 해를 보장하지만 실시간으로 환경이 변하는 다이나믹한 생산 환경(e.g. 작업의 투입, 기계 고장, 작업 시간의 변동성 등)에 적용하기에는 제한적이다. 즉, 환경에 대한 조건이 하나라도 변경된다면 처음부터 해를 다시 찾아야 한다는 단점이 존재하기 때문에 실제 생산 현장에서는 빠른 시간 안에 해를 찾을 수 있는 휴리스틱이 주로 사용되고 있다. 그러나 휴리스틱 또한 근시안적인 해법으로 의사결정 시점에서의 priority index를 최대화하는 방향으로 행동을

선택하기 때문에 미래에 대한 고려가 부재하다(Park et al. 2021).

표 1-2 Pros and Cons of Scheduling Methodologies

Methodology	Pros	Cons
Mathematical Programming, Branch and Bound algorithm	Guarantee the optimal solution	Curse of Dimensionality
Heuristic	Find solution within reasonable time	Not guarantee the optimal solution
Meta-Heuristic	Generate high quality solution	Difficult to apply at dynamic scheduling problem

하지만 강화학습 방법론은 변동성 있는 환경에서 효과적으로 적용이 가능하고, 누적 보상을 최대화함으로써 현재의 결정이 미래에 미치는 영향을 고려할 수 있기 때문에 두 방법론의 단점을 극복할 수 있다. 따라서 강화학습 방법론을 이용하여 동적 스케줄링 알고리즘을 개발하고자 하는 연구가 많이 수행되고 있다. [표 1-3]에서는 강화학습을 이용하여 병렬 기계 스케줄링 문제를 해결하고자 했던 선행연구를 문제의 조건과 함께 정리했다. Zhang et al. (2007)의 연구에서는 Q-learning 알고리즘을 적용하여 강화학습 에이전트가 병렬 기계 생산 시스템 내 작업들의 완료 상태 및 납기 정보, 그리고 각 기계의 작업 진행 상황 등의 정보를 입력 받아 다섯 가지 우선순위 규칙 중에서 mean weighted tardiness의 최소화를 위한 적절한 행동을 선택하도록 스케줄링 정책을 학습하였다. Zhang et al. (2012)은 상태 정보로서 병렬 기계 생산시스템 자체의 특성과 대상 작업들의 납기 관련 특성들을 포함하여 mean weighted tardiness의 최소화를 목적으로 하는 MDP 모델을 정의하였고 R-learning 알고리즘을 적용하여 작업 순서

결정을 위한 스케줄링 정책을 학습하였다. Yuan et al. (2016)의 연구에서는 불규칙한 기계 고장이 있는 병렬 기계 스케줄링 문제에서 최대 납기지연의 최소화와 지연이 발생한 작업의 수 최소화라는 두 가지 목적함수를 동시에 달성하기 위하여 Q-learning 알고리즘을 기반으로 세 가지 우선순위 규칙 중에서 적절한 행동을 선택하도록 스케줄링 정책을 학습하였다. Paeng et al. (2021)의 연구에서는 Deep Q-Networks(DQN) 알고리즘을 적용하여 에이전트가 total tardiness의 최소화를 목적으로 일정한 시간 간격마다 수행할 작업과 해당 작업을 할당할 기계를 선택하는 방식의 순서 의존적 set-up이 고려된 병렬 기계 스케줄링 알고리즘을 제안하였다. Julaiti et al. (2022)의 연구에서는 기계의 고장을 고려한 병렬 기계 스케줄링 문제를 생산 시스템의 통계적 정보에 기반하여 Partially Observable Markov Decision Process(POMDP)로 모델링하였다. 그리고 separate sampling 기법을 사용한 DDPG 알고리즘을 적용하여 에이전트가 세 가지 지표 (납기 지연 측면에서의 작업의 긴급도, 작업시간, 작업별 기계의 정상 작동시간 분포)에 대한 가중치를 조정하는 정책을 학습하도록 하였고, 학습된 정책을 기반으로 세 가지 지표를 가중합한 priority index를 생성하여 이에 따라 작업 순서를 결정하는 방식의 스케줄링 알고리즘을 제안하였다. Li et al. (2023)의 연구에서는 총 납기지연 최소화를 위하여 기계마다 작업 속도가 다른 병렬 기계 문제에서 Job Type 의존적 set-up이 고려된 문제를 Proximal Policy Optimization(PPO) 알고리즘을 이용하여 학습하였다. 의사결정 시점마다 크기가 달라지는 상태 정보를 고려하기 위하여 인공신경망으로는 Recurrent Neural Network(RNN), 그 중 Gated Recurrent Unit(GRU)를 사용하였고, 학습의 속도를 증가시키기 위하여 두 단계의 학습 과정을 사용하였다.

표 1-3 Related Works on PMSP using Reinforcement Learning

Author	Problem Type	Dynamic Components	Objective (Single / Multi)	Learning Algorithm	Industry Application
Paeng et al. (2021)	$R_m \mid S_{i,\mathrm{j}}, fmls \mid \Sigma T_i$	X	Minimize ΣT_i (SO)	Deep Q-learning	O (Semiconductor)
Li et al. (2023)	$Q_m \mid S_{i,j} \mid \Sigma T_i$	X	Minimize ΣT_i (SO)	Proximal Policy Optimization	X
Zhang et al. (2012)	$R_m \mid r_{i,j}, fmls \mid \Sigma w_i \times T_i$	Job insertion ($r_{i,j}$)	Minimize $\Sigma w_i \times T_i$ (SO)	R-learning	X
Julaiti et al. (2022)	$Q_m \mid bkdwn, fmls \mid \Sigma w_i \times T_i$	$random\ p_{i,j}$ Machine breakdown ($bkdwn$)	Minimize $\Sigma w_i \times T_i$ (SO)	Deep Deterministic Policy Gradient	X
Yuan et al. (2016)	$P_m \mid bkdwn \mid L_{max}, \Sigma U_i$	Machine breakdown ($bkdwn$)	Minimize L_{max}, ΣU_i (MO)	Q-learning	X
This Study (2023)	$P_m \mid S_{i,\mathrm{j}}\ r_{i,j} \mid \Sigma T_i, \Sigma S_{i,j}$	Job insertion ($r_{i,j}$) / $random\ p_{i,j}$	Minimize ΣT_i, $\Sigma S_{i,j}$ (MO)	Proximal Policy Optimization	O (Shipbuilding Industry)

1.2 연구 목적

대부분의 선행연구에서는 실제 산업의 생산 시스템이 아닌 개념적인 문제만을 대상으로 스케줄링 알고리즘의 유효성을 검증하였다. 또한 실제 생산 현장에서는 다양한 지표를 고려하여 현장 관리자가 스케줄링을 진행함에도 불구하고 단일 목적함수만을 가정한 선행연구가 대부분이었고, 다중 목적함수를 고려했던 Yuan et al. (2016)도 개념적인 문제에 대해서만 스케줄링 알고리즘을 학습하였다. 따라서 본 논문에서는 병렬 기계 문제에서의 셋업 최소화와 총 납기지연 최소화라는 다중 목적함수를 고려할 수 있는 스케줄링 알고리즘을 제안한다. 특히 조선소 내에서 용접 작업이 이루어지는 형강공장이라는 실제 시스템을 대상으로 두 가지 목적함수에 대해 성능이 좋다고 알려진 네 가지 우선순위규칙과 비교하였을 때, 본 논문에서 제안한 스케줄링 모델이 모든 목적함수를 달성하는 데 유효한 알고리즘이라는 것을 확인한다.

1.3 논문구성

본 논문은 총 8장으로 구성된다. 제 2 장에서는 본 연구에서 사용한 강화학습 방법론, 특히 Proximal Policy Optimization에 대하여 살펴본다. 또한 이산 사건 시뮬레이션을 기반으로 한 강화학습에 대하여 소개한다. 제 3 장에서는 대상 문제인 조선소의 형강공장에 대한 문제 정의와 가정, 그리고 목적함수에 대하여 정의한다. 제 4 장에서는 병렬 기계 스케줄링 문제를 강화학습으로 해결하기 위해 필요한 Markov Decision Process을 제시한다. 제 5 장은 제 3 장과 제 4 장에서 정의한 것을 토대로 학습한 결과를 검토할 것이다. 제 6 장은 제 5장에서 학습한 모델의 유효성을 검증하기 위하여 다른 우선순위 규칙과의 비교를 수행하고, 그 결과를 살펴본다. 제 7 장에서는 일반적인 병렬 기계 스케줄링 문제에 본 논문에서 제안한 스케줄링 알고리즘을 적용한다. 이는 조선소에 존재하는 다른 병렬 기계 스케줄링 문제에도 본 논문에서 제안한 스케줄링 알고리즘이 유효하게 적용할 수 있는 지 확인하기 위함이다. 마지막으로 제 8 장에서는 결론과 향후 연구방향을 제시한다.

제 2 장 방법론

2.1 강화학습

강화학습(Reinforcement Learning; RL)은 기계학습의 한 종류로, [그림 2-1]과 같이 에이전트가 환경과 상호작용을 하면서 순차적 의사결정 문제에 대한 최적의 정책을 학습하는 것을 목적으로 한다. 에이전트와 환경 간의 상호작용은 상태(s), 행동(a), 보상(r), 그리고 상태전이확률(p)로 구성된 수학적인 모델인 Markov Decision Process (MDP)로 표현할 수 있다. 에이전트는 이산적 시간 단계의 매 시점(t)마다 환경의 상태$(S_t \in \mathcal{S})$를 입력 받고, 정책(π)에 따라 행동$\left(A_t \in \mathcal{A}(\mathrm{s})\right)$을 선택한다. 이후 다음 시점$(t+1)$에서, 에이전트는 이전 행동$(A_t)$의 결과로서 보상$(R_{t+1})$을 받고 상태전이확률$(p)$을 통해 새로운 상태$(S_{t+1})$로 이동한다.

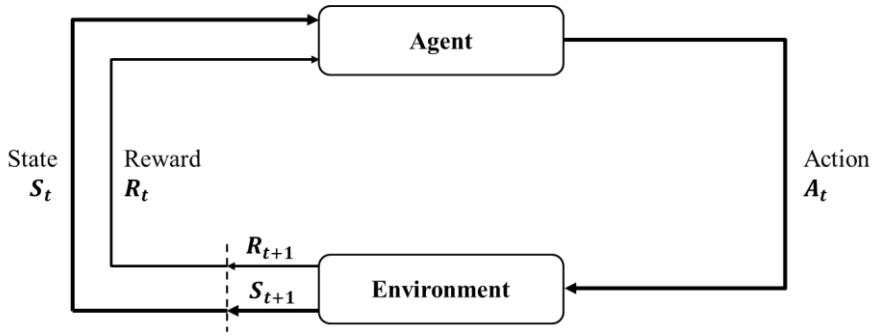


그림 2-1 The agent-environment interaction in a Markov Decision Process (Sutton and Barto 2018)

에이전트는 위와 같은 과정을 반복하면서 식 (1)과 같이 미래의 특정 기간 동안 자신이 받는 누적 보상으로 정의되는 반환값(G_t)이

최대가 되도록 정책 (π) 을 학습한다. γ 은 0과 1사이의 값을 갖는 감가율(discount ratio)로, 먼 미래에 받을 보상보다 가까운 미래에 받을 보상에 더 큰 가중치를 부여한다.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

정책 (π) 은 상태 s 가 주어졌을 때 가능한 행동을 선택할 확률로 정의된다. 이때 특정 상태 s 에서 주어진 정책 π 에 따라 행동했을 때 얻을 수 있는 반환값의 기대값을 상태가치함수(State-Value Function, $V_\pi(\mathrm{s})$)라고 하고, 식 (2)와 같이 표현할 수 있다. 식 (2)와 같이 현재 상태의 상태 가치 $V_\pi(s_t)$ 와 다음 상태에서의 상태가치 $V_\pi(s_{t+1})$ 사이의 관계를 나타낸 식을 벨만 방정식(Bellman Equation)이라고 한다. 마찬가지로 상태 s 에서 행동 a 를 선택한 이후 주어진 정책 π 에 따라 의사결정을 진행했을 때 얻을 수 있는 반환값의 기대값을 행동 가치함수(Action-Value Function, $Q_\pi(s,a)$)라고 하며, 식 (3)와 같이 나타낼 수 있다.

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(s_{t+1}) \mid S_t = s] \end{aligned} \tag{2}$$

$$\begin{aligned} Q_\pi(\mathrm{s},\mathrm{a}) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma \mathbb{E}_{a\sim\pi}[Q_\pi(s_{t+1}, a_{t+1})] \mid S_t = s, A_t = a] \end{aligned} \tag{3}$$

최적 정책(π_*)은 각 상태에서의 상태가치함수 값을 최대로 만드는 정책으로, 최적 정책 하에서의 가치함수를 최적 상태가치함수(Optimal Value Function, $V_*(s)$) 라고 하고 식 (4)로 나타낼 수 있다. 최적 상태가치함수와 마찬가지로, 상태 s에서 행동 a를 선택한 후 최적 정책 π_* 에 따라 의사결정을 진행했을 때의 행동가치 값을 최적 행동가치함수(Optimal Action−Value Function, $Q_*(s,a)$)라 하고, 식 (5)와 같이 나타낼 수 있다. 최적 상태가치함수 V_*와 최적 행동가치함수 Q_* 사이의 관계는 식 (6)과 같다. 여기서 최적 상태가치함수 V_*를 구하기 위한 벨만 방정식을 최적 벨만 방정식(Bellman Optimality Equation)이라고 한다. 최적 벨만 방정식은 최적 정책을 따르는 상태의 가치는 그 상태에서 선택할 수 있는 가장 좋은 행동으로부터 나오는 보상의 기댓값과 같다는 내용으로, 식 (7)으로 표현할 수 있다. 마찬가지로 최적 행동가치함수 Q_*를 구하기 위한 최적 벨만 방정식은 식 (8)와 같다.

$$V_*(s) = \max \mathrm{V}_\pi(\mathrm{s}) \tag{4}$$

$$Q_*(\mathrm{s},a) = \max Q_\pi(\mathrm{s},a) \tag{5}$$

$$Q_*(\mathrm{s},a) = \mathbb{E}[R_{t+1} + \gamma V_*(s_{t+1}) \mid S_t = s, A_t = a] \tag{6}$$

$$\begin{aligned} V_*(s) &= \max_{a\in\mathcal{A}} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s, A_t = a] \end{aligned} \tag{7}$$

$$Q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \tag{8}$$

강화학습에서 가치함수를 바탕으로 의사결정을 진행하는 방법을 가치기반 강화학습이라고 하며, 대표적인 예로는 Q-learning과 Deep Q-Networks (DQN)이 있다. Q-learning은 식 (8)을 기반으로 모든 상태-행동 조합에 대하여 최적 행동가치함수인 Q_* 를 학습하는 알고리즘으로, 식 (9)에 의해 Q 함수를 업데이트 하며 학습을 진행한다. Q-learning의 경우 가치함수를 학습하기 위한 타깃 정책으로는 탐욕정책을 사용하고, 행동을 선택하기 위한 행동 정책으로는 $\varepsilon-$탐욕정책을 사용하기 때문에 타깃 정책과 행동 정책이 다른 Off-policy 알고리즘에 해당한다.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)) \quad (9)$$

다만, Q-learning의 경우 상태-행동 조합에 대한 Q 함수의 값을 테이블 형식으로 저장하여 학습하기 때문에 상태와 행동 집합이 큰 문제에 적용하기에는 제한이 있었다. DQN 알고리즘은 Mnih et al. (2015)에서 제안된 알고리즘으로, 인공신경망을 통해 Q 함수의 값을 근사하기 때문에 고차원의 문제에도 적용이 용이하다는 장점이 있다. [그림 2-2]는 DQN 알고리즘의 전체적인 학습 과정을 나타낸다. 에이전트는 Q Network(가중치 θ)와 Target Q Network(가중치 θ^-)의 두 개의 인공신경망으로 구성되며, 환경은 Q Network와 상호작용하며 상태, 보상, 행동을 주고받으며 Replay Memory D에 상태, 행동, 보상, 다음상태(s_t, a_t, r_t, s_{t+1})를 저장한다. 이후 에이전트는 의사결정시점마다 Replay Memory D로부터 Sample을 추출하여 Target Q Network의 Q함수를 이용하여 [그림 2-2]의 Loss 함수를 계산해 Q Network의 가중치를 업데이트한다. 이후 주기적으로 Target Q Network의 가중치를 Q Network의 가중치로 업데이트 한다.

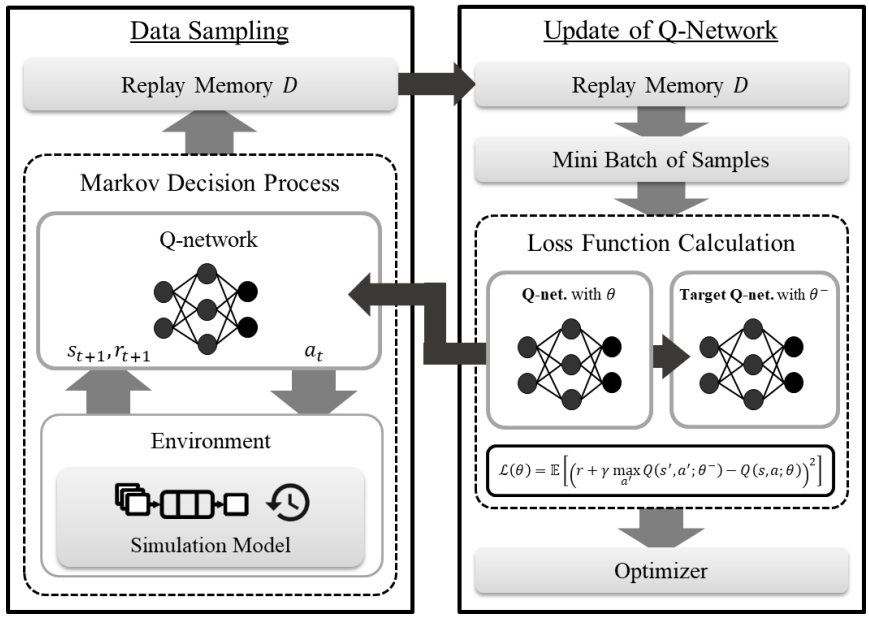


그림 2-2 Framework of DQN algorithm

강화학습에서 가치함수가 아닌 정책 자체를 평가하고 개선하는 방법을 정책기반 강화학습이라고 하며, 대표적인 방법으로는 Actor-Critic 방법이 있다. Actor-Critic은 Actor와 Critic의 두 가지 모듈을 가지고 있다. Critic 모듈은 환경으로부터 받은 보상을 이용해 이전 행동을 평가하며, Actor 모듈을 정책을 통해 행동을 선택한다.

Schulman et al. (2017)의 연구에서 제안된 Proximal Policy Optimization (PPO) 알고리즘 또한 Actor-Critic 구조를 활용한 정책기반 알고리즘에 해당한다. PPO 알고리즘은 식 (10)와 같이 정의한 대리 손실함수 (surrogate loss function) $L(\theta)$을 최소화하여 최적의 정책을 학습한다. PPO 알고리즘은 정책을 근사한 인공신경망의 가중치 θ를 업데이트 함에 있어 clip 함수를 도입함으로써 기존의 정책과 업데이트된 정책의 비율 $r_t(\theta)$가 $[1-\varepsilon, 1+\varepsilon]$의 범위 안의 값을 갖도록 제한하여 과도하게 정책이 업데이트 되는 것을 방지한다. $\hat{A}_t$는 advantage 함수로서 TD error δ_t를 사용하여 식 (11)와 같이 정의되는

한계 누적 보상 기대값(marginal expected sum of rewards)을 계산한다. [그림 2-3]는 PPO 알고리즘의 전체 학습 과정을 나타낸 그림이다.

$$L(\theta) = \hat{E}_t\left[max\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] \quad (10)$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (11)$$

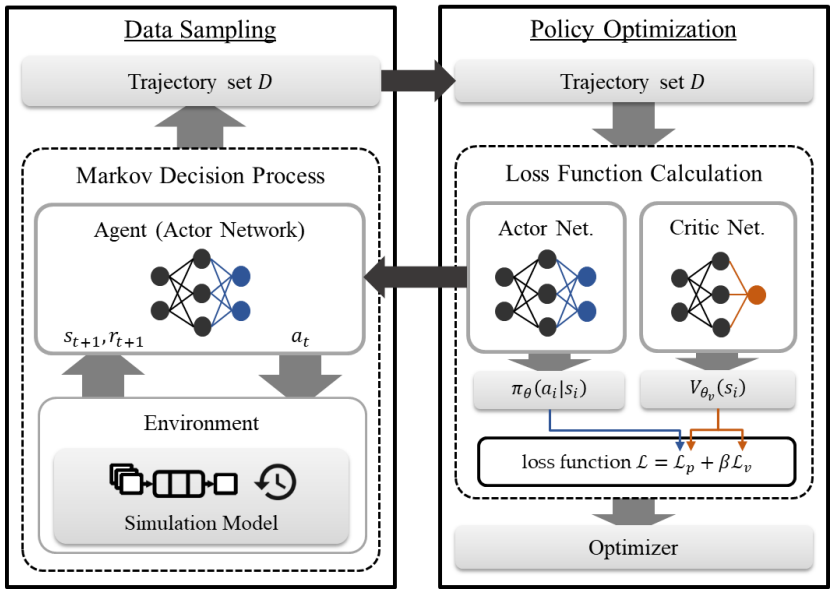


그림 2-3 Framework of PPO algorithm

2.2 시뮬레이션 기반 강화학습

이산 사건 시뮬레이션(Discrete Event Simulation; DES)는 이산 시스템의 거동을 분석하기 위한 방법으로, 제조/물류/교통/금융 등 다양한 분야에서 해석을 위한 도구로 많이 사용되어 오고 있기 때문에 DES를 위한 도구로서 다양한 종류의 상용 프로그램(Commercial Off-The-Shelf; COTS)들과 오픈소스 패키지들이 알려져 있다. Plant Simulation™, ARENA™, Anylogic™, Flexsim™ 등과 같은 상용 DES 프로그램들은 User Interface가 제공되어 DES의 시각화가 가능해 사용자가 직관적으로 이해할 수 있다는 장점이 존재한다. 다만, 구매 및 유지보수 비용과 기존 모델의 재사용성 및 확장성 측면에서는 한계가 존재한다. 반면 Java의 DESMO-J, C++의 SystemC, OMNet++, Python의 SimPy등의 오픈소스 기반의 DES 패키지는 User Interface가 지원되지 않거나, User Interface가 지원되는 패키지라고 하더라도 상용 소프트웨어보다 직관적이지 않다는 단점이 존재하지만, 비용이 존재하지 않는 점과 소스 코드에 대한 접근이 가능해 커스터마이징이 가능하다는 장점이 있다(Dagkakis and Heavey 2016). 따라서 수많은 연구들은 각각의 장단점을 바탕으로 DES tool을 선택하고, 연구에 활용하고 있다.

DES를 이용한 연구 중에는 DES 자체의 개발과 관련된 연구도 있지만, 인공지능에 DES를 활용한 연구들도 많이 진행되고 있다. 대표적인 사례로 제조 분야에서의 인공지능 연구를 진행함에 있어 제조 환경을 DES를 이용하여 모사하는 연구가 다수 존재한다. 인공지능의 한 종류인 강화학습에서도 환경을 모델링 함에 있어, DES를 적용한 사례를 다수 찾아볼 수 있다. [표 2-1]는 강화학습에 시뮬레이션을 사용한

사례이다. 최근 5년동안 DES 환경을 이용하여 대상 문제를 모델링하고 강화학습 알고리즘과 연결하여 문제를 해결하려고 한 사례가 다수 존재했다. 사용한 시뮬레이션 도구로는 사용자의 편의성에 따라 AnyLogic™, Plant Simulation™ 등 상용 프로그램을 사용한 연구가 다수 있었던 반면, 인공지능의 개발과 관련하여 많은 연구자들이 사용하고 있는 Python 언어로 개발된 오픈소스 패키지인 SimPy 패키지를 이용한 연구도 존재했다.

표 2-1 Related Researches on Reinforcement Learning using DES

DES tool (Category)	Author (year)	Interface	Target Problem
AnyLogic™ (COTS)	Thomas et al. (2018)	RL4J	Scheduling in manufacturing system
	Jang et al. (2018)	RL4F	Traffic signal control
	Pinciroli et al. (2020)	Pathmind	Energy system operation
Simio™ (COTS)	Greasley (2020)	Built-in RL function	Operation in manufacturing system
Plant Simulation™ (COTS)	Rabe et al. (2017)	MySQL	Logistic operation
	Shiue et al. (2018)	Built-in interface for C	Scheduling in manufacturing system
	Mayer et al. (2021)	TCP/IP	Operation in manufacturing system
Flexsim™ (COTS)	Pires et al. (2021)	Built-in interface for SQL	Operation in manufacturing system
	Preston (2017)	Built-in interface for Excel	Material handling plan
SimPy (Open Source)	Stricker et al. (2018)	Data transferred in Python	Scheduling in manufacturing system
	Menda et al. (2018)	Data transferred in Python	Bus and aircraft control
	Woo et al. (2021)	Data transferred in Python	Scheduling in manufacturing system

본 연구에서는 Stricker et al. (2018), Menda et al. (2018), Woo et al. (2021)와 같이 오픈소스 DES 패키지 중 하나인 SimPy 패키지를 기반으로 한 강화학습 방법론을 적용하였다. 해당 패키지는 다양한 인공지능 활용 패키지들이 개발되어 있는 Python 언어로 개발되었기 때문에 학습 알고리즘과의 연결이 용이하다. 또한 Pandas나 Numpy 등 고효율 데이터 처리 패키지와 연동할 수 있다는 점에서 데이터 처리 측면에서도 장점이 존재한다.

SimPy는 시뮬레이션의 환경과 시간 관리, 이벤트 관리 기능을 제공한다. 하지만 제조 환경을 모사하기에는 ① 시뮬레이션 객체간 이동과 ② 병렬 기계의 작업에 대한 기능이 부재했다. 따라서 본 연구에서는 Nam et al. (2022)에서 제안한 시뮬레이션 프레임워크를 바탕으로 [그림 2-4]와 같이 SimPy 패키지를 기반으로 제조 시뮬레이션에 필요한 기능을 개발한 뒤, 시뮬레이션을 학습 환경으로 하는 학습 알고리즘과 연결하였다.

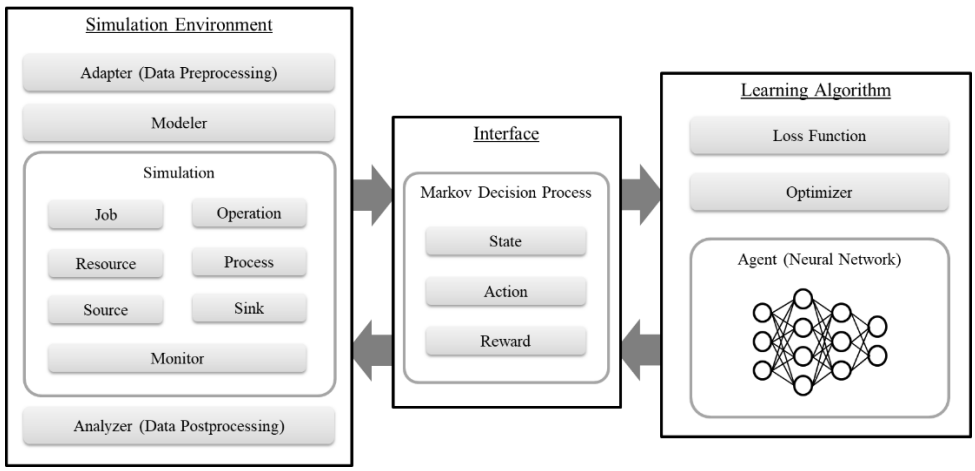


그림 2-4 Framework of simulation-based environment for reinforcement learning

시뮬레이션 시스템은 시뮬레이션에 필요한 데이터를 전처리하는 Adapter Module과 전처리 된 데이터를 시뮬레이션의 각 클래스로 모델링하는 Modeler Module, 제조환경을 모사한 Simulation Module, 시뮬레이션에서 기록된 이벤트를 바탕으로 필요한 지표를 계산하는 Analzer Module로 구성된다. 구체적으로 Simulation Module은, 작업의 객체가 되는 Job, 작업 스케줄인 Operation, 작업장 혹은 공장에 해당하는 Process, 작업에 필요한 리소스인 Resource, 객체를 생성하고 종료시키는 Source와 Sink, 마지막으로 시뮬레이션에서 발생하는 이벤트를 기록하는 Monitor의 7개의 클래스로 이루어져 있다.

Interface는 시뮬레이션 시스템과 학습 알고리즘을 연결하는 역할로, 시뮬레이션으로부터 필요한 정보를 바탕으로 MDP를 구성하는 상태, 보상을 계산하고 학습 알고리즘에 전달한다. 이후 학습 알고리즘으로부터 행동을 전달받아 시뮬레이션에 전달한다. 시뮬레이션은 Interface로부터 행동을 전달받아 상태변환확률 대신 다음 상태와 보상을 계산한다. 이러한 일련의 과정을 통해 시뮬레이션과 학습 알고리즘은 정보를 주고받으며 스케줄링 알고리즘을 학습하게 된다.

제 3 장 문제 정의

조선소 병렬 기계 문제로 모델링 할 수 있는 조선소 형강공장은 각종 블록의 제작에 필요한 보강재를 생산하는 공정으로, 후행공정에서 요구되는 물량을 적시에 공급하기 위해서는 효율적인 작업을 가능하게 하는 생산 계획 수립이 요구된다. 형강공장에서 생산되는 보강재는 T자형 보강재인 T-bar(T-shape section steel)로, [그림 3-1]과 같이 web과 face 부재의 용접을 통해 제작된다. 용접 공정에서는 보강재의 종굽힘 변형을 방지하기 위한 목적으로 web의 상단부에 고주파 유도 가열을 수행하게 되는데, web과 face의 두께 및 폭을 기준으로 고주파 유도 가열의 온도와 위치가 결정되고 이에 따라 용접 라인의 set-up이 변경된다.

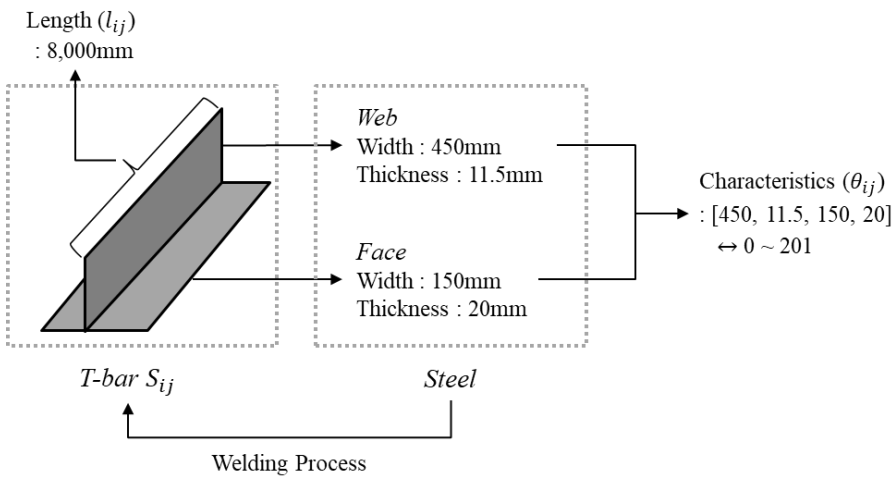


그림 3-1 Structure of Section Steel and Example of Characteristics

따라서 형강공정의 생산량 최대화는 동일한 스펙의 보강재들이 연속되도록 작업 순서를 계획하여 set-up 시간을 최소화함으로써

달성할 수 있다. 다만 각 보강재는 후행공정의 일정에 따라 납기일이 정해지기 때문에, 단순히 set-up을 최소화하는 것만 아니라 추가로 납기 준수의 관점에서 tardiness의 최소화도 고려해야 한다. 실제 현장에서는 set-up 시간 최소화를 위하여 같은 특성을 가지는 부재를 연이어 작업하기 위해 해당 부재가 입고될 때까지 의도적으로 용접 라인의 가동을 중지하기도 한다. 하지만 유휴 용접 라인이 있다는 것은 해당 시간 동안 입고가 되었음에도 작업이 되지 못하는 부재가 존재한다는 것을 의미하고, 대기 중인 부재의 납기 지연이 발생할 수 있다. 따라서 형강공정의 작업 순서 결정 문제는 두 목적함수가 상존하는 다목적 최적화 문제가 된다. 본 논문에서 대상으로 한 S 조선소의 형강공장에서는 보강재를 만들기 위한 web과 face가 크레인을 통해 세 개의 병렬 작업 라인 중 하나로 이동되어 배제 및 용접 공정을 수행한다. 최종적으로 용접이 완료된 보강재는 사상 공정을 수행하고 같은 블록에 속하는 보강재들은 파렛트 단위로 적치되어 반출된다. 실제 S 조선소의 실적 데이터를 분석한 결과, 한 주에 약 80개의 블록에 속하는 부재들이 작업된다.

가장 상위 개념인 블록에 대한 데이터 구조는 [표 3-1]과 같다. 블록 집합 $B = \{B_1, \dots, B_N\}$에 속하는 블록 $B_i (i = 1, \dots, N)$은 n_i개의 보강재로 이루어져 있으며, 납기일 d_i를 가진다. 하나의 블록에 속하는 부재들(web과 face)은 선행 공정에 해당하는 절단 공정이 완료된 후 도착시간 간격(Inter Arrival Time; IAT)을 가지고 용접 작업을 위한 대기열에 도착한다. 도착시간 간격 분포는 식 (12), (13)와 같다. 블록 B_i의 납기일 d_i은 블록 B_i가 절단 공정을 마친 후 용접 라인에 도착하는 시간인 r_i와 due date tightness (τ), 그리고 블록 B_i에 속하는 부재들의 평균 작업 시간의 합으로 정해진다. 블록 B_i의 납기일 d_i는 식 (14)와 같이 계산된다. 블록 B_i에 속하는 보강재의 집합을 $S_i = \{S_{i1}, \dots, S_{in_i}\}$에

속하는 보강재 $S_{ij}(j=1,\dots,n_i)$는 동일한 부재의 개수인 n_{ij}, web과 face의 두께와 폭 정보에 해당하는 특성인 θ_{ij}, 용접 두께인 t_{ij}, 보강재의 용접 길이인 l_{ij}를 속성으로 가진다. 실제 데이터의 예시는 [표 3-1]에 나타냈다. 그 중 set-up 시간을 결정 짓는 θ_{ij}는 4개의 실수형 자료의 벡터 형태로 구성되어 있는데, 학습에서는 [그림 3-1]과 같이 0부터 201 사이의 정수형으로 변환시켜 스칼라 형태로 입력하였다.

표 3-1 Data Structure of Profile Shop

Block	Section Steel	Characteristics
B_i ($n_1=10$, $r_i=3$)	S_{i1}	$n_{i1}=5\ [EA]$ $\theta_{i1}=5\ (\leftrightarrow (200,15,150,15)[mm])$ $t_{i1}=4.5\ [mm]$ $l_{i1}=8{,}800\ [mm]$ $v_{i1}=1{,}100\ [mm/min]$ $\overline{p}_{i1}=8\ [min]$
	S_{i2}	$n_{i2},\theta_{i2},t_{i2},l_{i2},v_{i2},\overline{p}_{i2}$
	⋮	
	S_{i10}	$n_{i10},\theta_{i10},t_{i10},l_{i10},v_{i10},\overline{p}_{i10}$

본 연구에서 보강재의 작업은 동일한 생산 능력을 갖는 3개의 병렬 용접 라인 중 하나에서 수행된다고 가정한다. 보강재 S_{ij}에 대한 용접 공정 속도 v_{ij}는 보강재의 용접 두께 t_{ij}에 의해 식 (15)로 결정된다. 따라서 보강재 S_{ij}의 평균 작업시간 $\overline{p}_{ij}$은 식 (16)와 같이 보강재의 길이 l_{ij}를 공정속도 v_{ij}로 나눈 결과가 된다. 하지만 실제 현장에서는 작업 시간에 있어 변동성이 존재하기 때문에, 본 연구에서는 식 (16)로 계산된 평균 작업 시간을 사용하여 식 (17)과 같이 정의한 균등분포(uniform distribution)와 작업 시간의 변동성 지수 δ_{pt}에 따라 보강재 S_{ij}의 작업 시간이 확률적으로 결정된다고 가정한다.

$$Inter\ Arrival\ Time\ (IAT) \sim exponential\ \left(\frac{1}{\lambda}\right) \tag{12}$$

$$\frac{1}{\lambda} = \frac{960[min/day] \times 6[day/week]\ \times n[week]}{N} \tag{13}$$

$$d_i[day] = \left\lfloor r_i[day] + \tau \times \sum_{j=1}^{n_i} \overline{p}_{ij}[day] \times n_{ij} \right\rfloor \tag{14}$$

$$v_{ij} = 1200 - \frac{t_{ij} - 4.5}{0.5} \times 50\ [mm/min] \tag{15}$$

$$\overline{p}_{ij} = \frac{l_{ij}}{v_{ij}}\ [min] \tag{16}$$

$$p_{ij} \sim U\left((1 - \delta_{pt}) \times \overline{p}_{ij}, (1 + \delta_{pt}) \times \overline{p}_{ij}\right) \tag{17}$$

형강공정의 용접 라인에 대하여 본 연구에서 추가적으로 가정한 사항은 다음과 같다. 부재의 용접을 담당하는 용접 라인은 총 세 개이며, 각 용접 라인은 한 번에 하나의 보강재만 작업할 수 있다. 이때 용접 라인은 일주일 중 일요일을 제외한 6일 동안 가동되며, 하루 작업 가능 시간은 16시간이다. 그리고 각 용접 라인에서 연속해서 작업되는 두 보강재의 web과 face의 값(부재의 특성, θ_{ij})이 다르면 set-up 변경이 이루어지며, set-up 시간은 작업 순서와 상관없이 5분이라고 가정한다. 위 가정 사항들을 정리하면 다음과 같다.

- 세 개의 용접라인을 통해 용접작업을 진행함
- 각 용접 라인은 동일한 생산능력을 가짐
- 각 용접 라인은 한 번에 하나의 부재만 작업 가능함
- 작업은 하루 16시간씩 주 6일 작업함
- 두 보강재의 특성(θ_{ij})이 다르면 5분의 set-up 시간이 발생함

입력 데이터와 가정사항을 바탕으로 실제 조선소 형강공장의 거동을 모사한 이산 사건 시뮬레이션의 흐름은 [그림 3-2]와 같다.

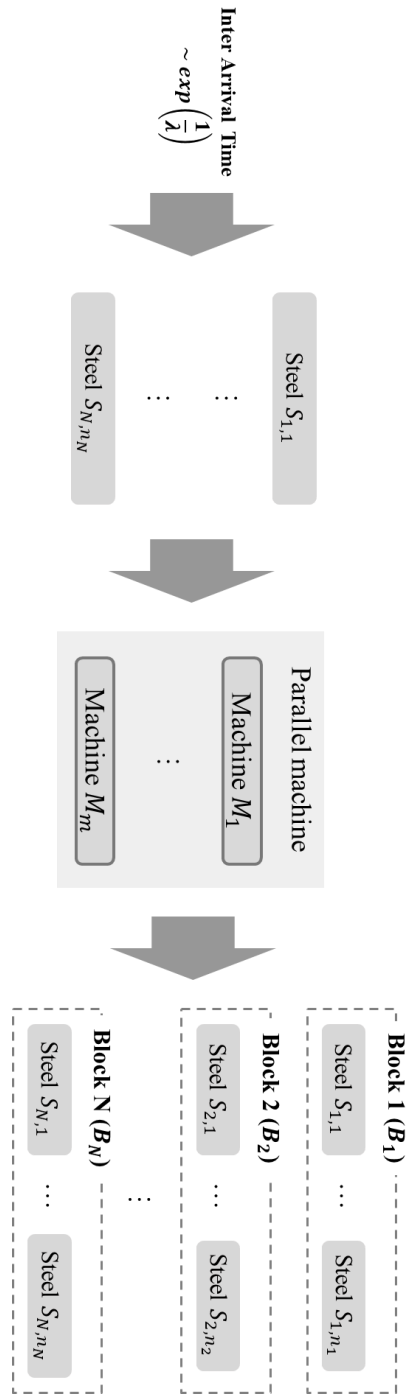


그림 3-2 Simulation Framework of Profile Shop

위와 같은 병렬 용접 라인에서 스케줄링 알고리즘의 결정 변수는 각 강재의 작업 시작 시간과 작업이 이루어질 용접 라인이다. 이를 통해 본 논문에서 개발한 스케줄링 알고리즘이 달성하고자 하는 목적함수는 납기 지연 시간의 최소화와 부재가 용접 라인에서 작업할 때 발생하는 셋업 시간의 최소화이다. 목적 함수와 관련된 생산 지표 중 하나인 납기 지연은 블록 별로 계산되는 값으로, 블록 B_i에 속하는 모든 부재들의 작업이 종료된 시간 C_i와 블록 B_i의 납기일 d_i의 차로 계산된다(식 (18)). 또 다른 생산 지표인 셋업 시간은 부재 S_{ij}가 용접 라인 M_k에서 작업될 때 부재의 특성(θ_{ij})과 용접 라인 M_k의 세팅 값(θ_k)이 다르면 발생하는 셋업 시간(σ_{ijk})이다(식 (19)). 해당 지표를 이용하여 두 목적함수를 식으로 표현하면 식 (20), (21)과 같다.

$$T_i\,[day] = max(0, C_i - d_i) \tag{18}$$

$$\sigma_{ijk}\,[min] = \begin{cases} 0, & \theta_{ij} = \theta_k \\ 5, & \theta_{ij} \neq \theta_k \end{cases} \tag{19}$$

$$Minimize \sum_{i=1}^{N} T_i \tag{20}$$

$$Minimize \sum_{i=1}^{N} \sum_{j=1}^{n_i} \sigma_{ijk} \tag{21}$$

제 4 장 모델링

조선소 형강공정에 대한 작업 순서 결정 문제에 강화학습 방법론을 적용하기 위해서는 주어진 문제를 MDP로 모델링 해야한다. MDP는 상태(s), 행동(a), 보상(r), 그리고 상태변환확률(p)의 네 가지 요소로 에이전트와 환경의 상호작용을 정의한다. 본 연구에서는 각 용접라인에서 보강재의 작업이 완료된 순간을 의사결정 시점으로 정의하였다. 에이전트는 환경으로부터 해당 시점의 상태를 입력 받아 행동을 결정한다. 또한 학습 환경을 이산 사건 시뮬레이션 모델로 구현함으로써 환경에서는 에이전트가 선택한 행동을 바탕으로 상태변환확률 대신 시뮬레이션을 통해 다음 상태와 보상을 계산한다. 이러한 과정으로 에이전트는 샘플 데이터를 획득하고 학습 알고리즘을 통해 최적 정책을 학습한다. [그림 4-1]은 형강공정 스케줄링 문제에 대한 전체적인 학습 프레임워크이다.

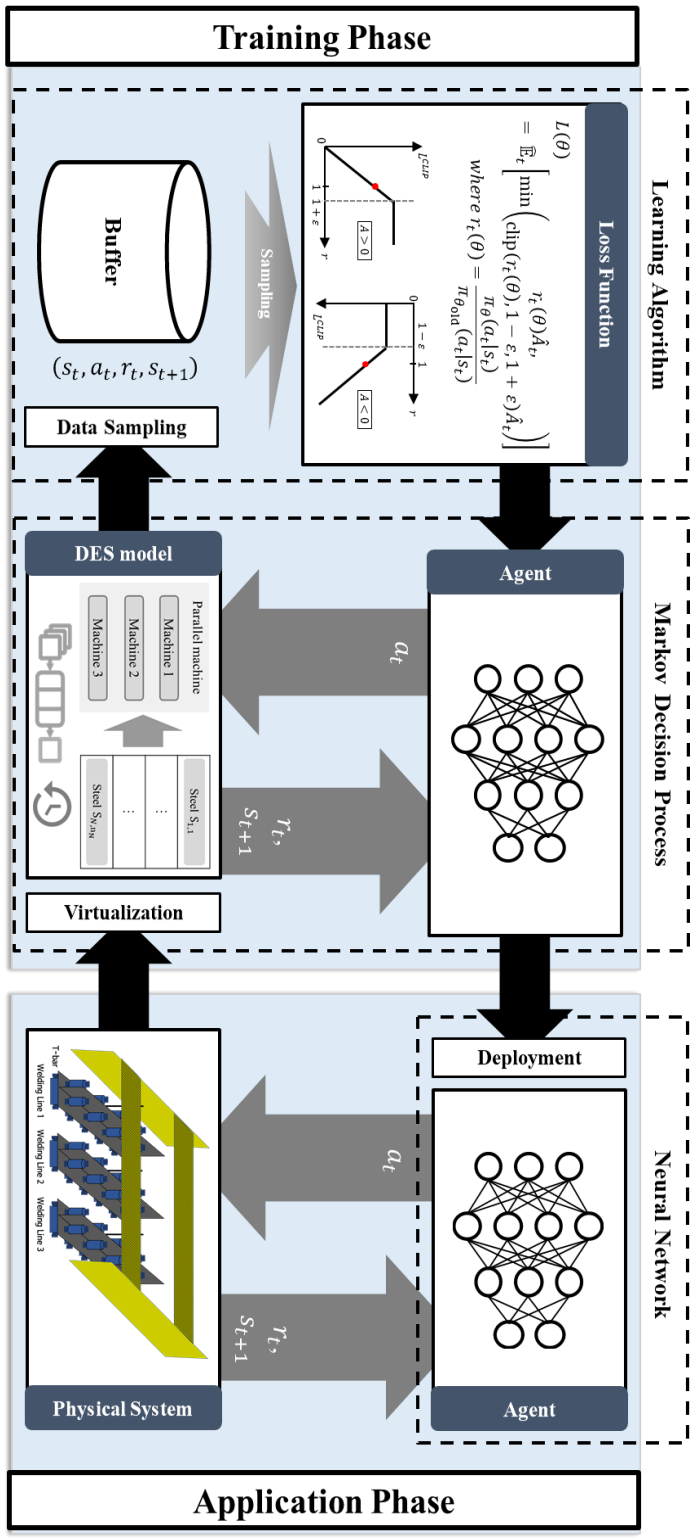


그림 4-1 Overall Learning Framework

4.1 상태

에이전트의 행동 결정 기준이 되는 상태는 각 의사결정 시점 t 에서 작업이 완료되지 않은 보강재들과 각 용접 라인에 대한 정보를 기반으로 총 네 가지의 특성벡터로 구성된다. 그리고 대상 계획 기간이 달라져도 이미 학습된 에이전트의 적용이 가능하도록 상태의 크기가 블록의 개수와 독립적인 상태벡터를 정의했다. 상태의 크기는 $2m+8$이다.

첫 번째 특성벡터(f_1)는 set-up 시간 최소화와 관련된 정보로서, 식 (22)와 같이 정의된다. 식 (22)는 의사결정 시점에서 작업되지 않은 부재 $\left(Q(t)\right)$ 중 해당 시점에서 각 용접 라인의 set-up value (θ_k) 와 동일한 특성 $\left(\theta_{ij}\right)$을 갖는 부재의 비율이다. 즉, 어떤 부재가 용접 라인 M_k로 할당되었을 때 셋업 변경 없이 작업할 수 있을 확률이다. 해당 비율은 각 용접라인마다 계산하기 때문에 f_1의 크기는 용접 라인의 수인 m이다.

$$f_{1,k}=\begin{cases}\dfrac{N_k}{N_q} & If\ Machine\ M_k's\ set-up\ is\ detemined\\ 1.0 & Otherwise\ (Initial\ State)\end{cases},$$

$$(k=1,\dots,m) \tag{22}$$

N_k : $the\ number\ of\ steels\ that\ satisfy\ condition\ \theta_k = \theta_{ij}$

N_q : $the\ number\ of\ steels\ in\ Q(t)$

두 번째(f_2)및 세 번째(f_3)특성벡터는 tardiness 최소화 및 set-up 최소화와 관련하여, 작업 대기 중인 부재에 대한 정보로, 식 (23)와 같이 정의된다. f_2 는 각 용접라인의 set-up value (θ_k) 와 동일한 특성 (θ_{ij}) 을 갖는 부재를 대상으로, f_3 는 θ_k 와 θ_{ij} 가 다른 부재를 대상으로 하여 각 부재의 Tardiness level을 계산하였고, Zhang et al. (2012)에서 제안된 tightness를 식 (24), (25)와 같이 일부 변경하여 본 연구에 적용하였다. Tardiness level은 의사결정 시점에서 작업 대기 중인 부재가 속한 블록의 남은 예상 작업 시간을 고려했을 때의 예상 지연 발생 정도로 구분하였고, [표 4-1]에 정리하였다. 구체적으로 작업 시간의 변동성을 고려하여 계산된 최대 작업시간에 따라 작업이 진행되어도 지연이 발생하지 않는 경우(Level 1), 최대 작업시간을 고려하면 지연이 발생하지만, 최소 작업시간에 따라 작업이 진행되면 지연이 발생하지 않는 경우(Level 2), 최소 작업시간으로 진행되어도 지연이 예상되지만 아직 지연이 발생하지 않은 경우(Level 3), 이미 지연이 발생한 경우(Level 4)로 Tardiness level을 구분하였다. f_2 와 f_3의 크기는 각각 Tardiness level의 개수인 4이다.

$$
\begin{aligned}
&f_{2(3),\,g} = \frac{N_{2(3),g}}{N_{w,\,2(3)}} \quad (g = 1,\ 2,\ 3,\ 4)\\
&N_{2(3),g} : the\ number\ of\ Steels\ which\ has\ tardiness\ level\ g\\
&N_{q,\,2(3)} : the\ number\ of\ Steels\ which\ have\ not\ been\ worked\ yet\\
&\qquad that\ satisfy\ condition\ \theta_k = \theta_{ij}(f_2)\ or\ \theta_k \neq \theta_{ij}\ (f_3)
\end{aligned}
\tag{23}
$$

$$
Tardiness\ Level\ g = \begin{cases} 1, & if\ d_i - t \in (max\, r_{ij},\ +\infty) \\ 2, & if\ d_i - t \in (min\, r_{ij}, max\, r_{ij}] \\ 3, & if\ d_i - t \in (0, min\, r_{ij}] \\ 4, & if\ d_i - t \in (-\infty,\ 0] \end{cases} \tag{24}
$$

$$max\, r_{ij} = \sum_{j=1}^{N_{r,i}} (1 + \delta_{pt}) \times \overline{p}_{ij} \times n_{ij}$$

$$min\, r_{ij} = \sum_{j=1}^{N_{r,i}} (1 - \delta_{pt}) \times \overline{p}_{ij} \times n_{ij} \quad (25)$$

$N_{r,i}$: $the\ number\ of\ remaied\ Steels\ in\ Block\ B_i$

표 4-1 Description of Tardiness Level

Tardiness Level g	Description
1	No tardiness
2	It might have tardiness
3	It must have tardiness
4	It starts with tardiness

네 번째 특성벡터 (f_4)는 용접 라인에 대한 일반적인 정보로서, 의사결정 시점을 기준으로 각 용접라인의 작업 진도율이다. f_4 는 식 (26)와 같이 정의된다. f_4의 크기는 용접 라인의 수인 m이다.

$$f_{4,k} = \begin{cases} \frac{t_{s,ij,k} + \overline{p}_{ij} - t}{\overline{p}_{ij}}, & Machine\ M_k\ is\ working \\ 0, & Machine\ M_k\ is\ idle \end{cases}, \quad (k = 1, \dots, m) \quad (26)$$

$t_{s,ij,k}$: $the\ start\ time\ of\ Steel\ S_{ij}\ in\ Machine\ M_k$

4.2 행동

에이전트가 다음으로 작업할 부재를 직접 선택하는 방식으로 행동을 정의할 경우, 투입 시간 간격과 작업으로 인해 각 의사결정 시점에서의 선택 가능한 행동 집합의 크기가 달라지는 문제가 발생한다. 따라서 에이전트의 선택 가능한 행동은 의사결정 시점 t에서의 작업 가능한 부재들의 집합인 $Q(t)$에서 다음으로 작업할 부재를 선택하기 위한 우선순위 규칙을 선택하는 것으로 정의했다. 각 우선순위 규칙들은 tardiness 또는 set-up 발생 최소화 문제를 풀기 위하여 고안된 대표적인 우선순위규칙들로, SSPT(Shortest processing time and Shortest setup time), ATCS(Apparent Tardiness Cost with Setup), MDD(Modified Due Date), COVERT(Cost OVER Time) rule로 구성된다.

SSPT rule은 SPT(Shortest Processing Time) rule과 SST(Shortest Setup Time) rule을 결합한 우선순위 규칙으로, 식 (16)을 통해 계산된 평균 작업시간 $\left(\overline{p}_{ij}\right)$과 식 (19)을 통해 계산된 set-up 시간 $\left(\sigma_{ijk}\right)$을 더해서 계산된 값이 작을수록 높은 우선순위를 부여한다. SSPT rule에서 의사결정 시점 t에서의 용접라인 M_k를 위한 부재 S_{ij}는 식 (27)을 통해 결정된다.

$$S_{ij} = argmin\left\{\sigma_{ijk} + \overline{p}_{ij}, \forall\, S_{ij} \in Q(t)\right\} \tag{27}$$

ATCS rule은 total weighted tardiness의 최소화를 목적으로 제안된 ATC(Apparent Tardiness Cost) rule을 set-up 시간 최소화도 고려하도록 수정한 우선순위규칙이다(Lee and Pinedo 1997). ATCS

rule은 짧은 작업시간과 적은 slack time, 그리고 짧은 set-up time을 갖는 작업일수록 높은 우선순위를 부여한다. ATCS rule에서 의사결정 시점 t에서의 용접라인 M_k를 위한 부재 S_{ij}는 식 (28), (29), (30)을 통해 결정된다. 본 연구에서 두 파라미터 k_1, k_2 는 Lee and Pinedo (1997)에서 제안한 식 (31), (32), (33)과 시뮬레이션을 통해 [표 4-2]의 순서에 따라 실험적으로 결정하였다.

$$S_{ij} = argmax\left\{ \frac{1}{\bar{p}_{ij}} exp\left(-\frac{max\left(d_i - \bar{p}_{ij} - t, 0\right)}{k_1 \bar{p}}\right) exp\left(-\frac{\sigma_{ijk}}{k_2 \overline{\sigma}}\right), \right\} \quad \forall S_{ij} \in Q(t) \tag{28}$$

$$\overline{p} = \frac{1}{N_q} \sum^{N_q} \overline{p}_{ij} \tag{29}$$

$$\overline{\sigma} = \frac{1}{N_q} \sum^{N_q} \sigma_{ijk} \tag{30}$$

표 4-2 Determination of parameters of ATCS rule

Step 1	Calculate $\eta = \frac{\overline{\sigma}}{\overline{p}}$, $\mu = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n_i} n_{ij}}{m}$	(31)
Step 2	Simulation and Get $\max d_i$, $\min d_i$, $\overline{d}$, C_{max}	
Step 3	Calculate $\mathrm{R} = \frac{d_{max} - d_{min}}{C_{max}}$, $\tau = 1 - \frac{\overline{d}}{C_{max}}$, $\mathrm{A}_2 = \begin{cases} 1.8, & \tau < 0.8 \\ 2.0, & \tau \geq 0.8 \end{cases}$	(32)
Step 4	Determine k_1, k_2 with $k_1 = 1.2 \ln(\mu) - R, \quad k_2 = \frac{\tau}{A_2 \sqrt{\eta}}$	(33)

MDD rule은 EDD(Earliest Due Date) rule과 SRPT(Shortest Remaining Processing Time) rule을 결합한 우선순위 규칙으로, 납기일과 남은 작업 시간을 고려한 예상 작업 완료 시점이 빠른 작업일수록 높은 우선순위를 부여한다. MDD rule에서 의사결정 시점 t에서의 용접라인 M_k를 위한 부재 S_{ij}는 식 (34)을 통해 결정된다

$$S_{ij} = argmin\left\{max\left(d_i, t + \overline{p}_{ij}\right), \forall\, S_{ij} \in Q(t)\right\} \tag{34}$$

COVERT rule은 작업 시간이 짧고 추정 대기 시간 대비 slack time의 비율이 작은 작업에 우선순위를 부여한다. COVERT rule에서 의사결정 시점 t에서의 용접라인 M_k를 위한 부재 S_{ij}는 식 (35)에 의해 결정된다. 본 연구에서 파라미터 k 는 [0.1, 20]의 범위에서 0.1 간격으로 k값을 달리하여 case study를 진행했을 때, mean tardiness가 가장 작은 결과를 보인 k값으로 결정하였다.

$$S_{ij} = argmax\left\{\frac{1}{\bar{p}_{ij}} max\left(1 - \frac{max\left(d_i - \bar{p}_{ij} - t, 0\right)}{k\bar{p}_{ij}}, 0\right), \right\} \quad \forall\, S_{ij} \in Q(t) \tag{35}$$

4.3 보상

강화학습에서 보상은 학습의 성능을 좌우하는 요소로, 목적함수와 직접적으로 관련이 있다. 본 논문에서 다루는 문제는 tardiness 최소화와 set-up 시간 최소화라는 두 목적함수를 고려하는 다목적 최적화 문제로 보상은 tardiness 최소화와 set-up 시간 최소화의 두 가지 관점을 모두 포함하도록 정의되어야 한다.

Tardiness 최소화 관점에서의 보상(r_T)은 식 (37)과 같이 정의된다. 의사결정 시점 사이에 모든 부재의 작업을 마친 블록들이 존재한다면, 각 블록의 Tardiness를 식 (36)과 같이 정의된 지수함수를 통해 [-1, 0] 범위 안으로 변환한 후, 블록 별 penalty $(r_{T,i})$의 합을 보상으로 정의하였다. 즉, 각 블록 별로 발생한 tardiness가 클수록 최대 penalty인 -1에 가까운 보상을 할당한다. Tardiness에 따른 penalty 그래프는 그림 [4-1]와 같다.

$$r_{T,i} = e^{max(C_i - d_i, 0)} - 1 \tag{36}$$

$$r_T = \sum r_{T,i} \tag{37}$$

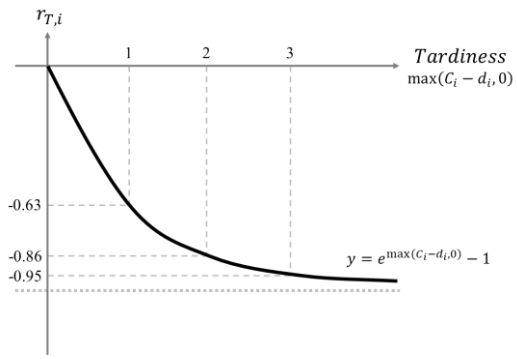


그림 4-2 Example of r_T

Set-up 시간 최소화 관점에서의 보상 (r_S) 은 식 (38)와 같이 정의된다. 에이전트가 선택한 우선순위 규칙에 따라 용접 라인 M_k에서 다음으로 작업할 부재 S_{ij}를 선택하여 작업하였을 때, set-up 시간이 발생하면 −0.1의 penalty를, set-up 시간이 발생하지 않는다면 0의 보상을 할당하였다. 이 때 r_T는 블록 단위로 발생하는 penalty이고 r_S는 부재 단위로 발생하는 penalty이기 때문에 두 보상의 발생 횟수 간의 불균형이 존재한다. 두 보상 간의 scaling을 위하여 r_S의 경우 r_T의 최소값인 −1의 10분의 1인 −0.1의 penalty를 부여하였다.

$$r_S = \begin{cases} -0.1, & \theta_k \neq \theta_{ij} \\ 0, & \theta_k = \theta_{ij} \end{cases} \tag{38}$$

최종 보상은 식 (39)와 같이 두 보상 간의 가중치를 6대 4 비율로 한 가중합으로 계산된다.

$$r_t = 0.6 \times r_T + 0.4 \times r_S \tag{39}$$

제 5 장 학습

조선소의 현장 데이터를 분석한 결과, 조선소 형강공정의 용접라인에서는 일주일 기준 약 80개의 블록에 대한 보강재 물량(약 700개)이 작업되는 것을 확인할 수 있었다. 따라서 에이전트의 학습 시나리오는 일주일에 해당하는 물량인 블록 80개에 포함된 보강재를 세 개의 용접라인에서 작업할 때의 작업 순서를 결정하는 문제로 설정했다. 즉, 제 3 장에서의 N과 m을 각각 80과 3으로 설정하였다.

또한 학습된 정책의 일반화 성능을 위하여 본 논문에서는 조선소로부터 획득한 총 754개의 블록 데이터를 매 에피소드마다 80개의 블록 데이터를 랜덤으로 샘플링하여 에이전트의 학습을 진행하였다. 이 때 각 블록의 납기일은 일요일을 제외하고 0에서 5사이의 정수값으로 인코딩된 월요일에서 토요일 중에서 랜덤으로 배정했다.

학습 알고리즘으로 사용한 PPO 알고리즘의 pseudocode는 [표 5-1]과 같다. PPO 알고리즘을 통해 에이전트가 학습할 정책은 인공신경망으로 모델링 되며, 본 논문에서는 총 5개의 완전 연결계층(fully-connected-layer)으로 인공신경망을 구성한다. 입력층은 상태 벡터의 크기와 동일한 14개의 노드를 갖고, 세 개의 은닉층은 차례로 512개, 512개, 256개의 노드를 갖는다. 마지막으로 출력층은 전체 행동의 수와 동일한 4개의 노드를 갖는다. 이 때 마지막 출력층을 제외한 첫 번째에서 네 번째 층은 모두 활성함수로 ReLU (Rectified Linear Unit)함수를 사용한다.

표 5-1 Pseudocode of PPO algorithm

Proximal Policy Optimization algorithm

Initialize weights θ of the actor network π_θ and weights θ_v of the critic network V_{θ_v}

for episode = $1, \ldots, E$ **do**

 # Data Sampling

get initial state s_0

repeat

calculate the probabilities $\pi_{\theta_{\text{old}}}(\cdot|s_t)$ for all actions

select an action a_t according to $\pi_{\theta_{\text{old}}}(\cdot|s_t)$

execute action a_t in DES simulator and observe reward r_t and next state s_{t+1}

put sample (s_t, a_t, r_t, s_{t+1}) into trajectory set D

$t \leftarrow t+1$ and $s_t \leftarrow s_{t+1}$

until s_{t+1} is terminal

 # Policy Optimization

for epoch = $1, \ldots, K$ **do**

calculate the generalized advantage estimates $\hat{A}_i$ for each sample in D

$\hat{A}_i = \delta_i + (\gamma\lambda)\delta_{i+1} + \cdots + (\gamma\lambda)^{T-i+1}\delta_{T-1}$

where $\delta_i = \begin{cases} r_i + \gamma V_{\theta_v}(s_{i+1}) - V_{\theta_v}(s_i) & if\, s_{i+1}\ is\ non-terminal \\ r_i - V_{\theta_v}(s_i) & otherwise \end{cases}$

calculate loss function $\mathcal{L}_p$ for the actor network

$\mathcal{L}_p = -\frac{1}{|D|}\sum_{i=1}^{|D|} \min\left(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta),\ 1-\varepsilon,\ 1+\varepsilon)\hat{A}_i\right)$

where $r_i(\theta) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}$

calculate loss function $\mathcal{L}_v$ for the critic network

$\mathcal{L}_v = \frac{1}{|D|}\sum_{i=1}^{|D|} {\delta_i}^2$

optimize θ, θ_v using Adam optimizer with loss function $\mathcal{L} = \mathcal{L}_p + \beta\mathcal{L}_v$

end for

empty the trajectory set D

end for

PPO 알고리즘의 하이퍼파라미터 설정을 위하여 Learning Rate α를 [0.005, 0.001, 0.0005, 0.0001] 범위에서, Optimization Epoch K를 1또는 2로 달리한 총 8개의 하이퍼파라미터 셋에 대해 실험을 진행하였고, 가장 높은 성능을 보인 조합을 사용하였다([표 5-2]). 에이전트는 총 10,000번의 에피소드동안 학습을 수행한다. 이 때 각 에피소드에서의 학습은 에이전트가 한 에피소드 동안 환경과의 상호작용을 통해 획득한 샘플을 바탕으로 각 에피소드가 끝날 때 1번의 가중치 업데이트가 이루어진다. 가중치 업데이트 시 손실함수 값은 Clipping Parameter ε, Discount Ratio γ, GAE Parameter λ를 각각 0.2, 0.98, 0.95로 설정하여 계산된다. 최적화 알고리즘으로는 Learning Rate α를 0.0001로 설정한 Adam 알고리즘을 적용한다.

표 5-2 Hyper-Parameters in the Learning Phase for PMSP of Shipbuilding

Hyper-parameter	Value
Number of Episodes E	10,000
Optimization Epoch K	1
Clipping Parameter ε	0.2
Discount rate γ	0.98
GAE Parameter λ	0.95
Learning Rate α	0.0001

마지막으로 형강공정의 거동을 모사한 시뮬레이션에 사용한 파라미터는 [표 5-3]와 같다. 에피소드마다 스케줄링 대상 블록 수(N)는 80개이며, 총 세 개의 용접라인(m)에서 작업된다. 블록의 평균 투입 간격 $\left(\frac{1}{\lambda}\right)$ 은 72분이며, 각 블록이 용접라인을 위한 대기열로 입고되는 요일(r_i)은 월요일에서 토요일 사이에서 무작위로 배정한 후,

0에서 5사이의 값으로 변환하였다. 각 블록의 납기일을 결정짓는 due date tightness τ 는 최소 0.8, 최대 1.2의 값을 갖는 uniform distribution을 통해 결정된다. 마지막으로 작업 시간의 변동성 지수인 δ_{pt}는 최소 0.1, 최대 0.5의 값을 가지는 uniform distribution에 의해 결정된다. 마지막으로 네 개의 행동 중 ATCS rule의 파라미터는 [표 4-2]에 의해 $k_1 = 5.953,\ k_2 = 1.0$ 으로 결정하였다. 마찬가지로 COVERT rule의 파라미터는 case study를 통해 $k = 6.6$으로 결정하였다.

표 5-3 Simulation Parameters in the Learning Phase for PMSP of Shipbuilding

Parameter	Value
The number of Blocks N	80
The number of Machines m	3
Average of IAT $\frac{1}{\lambda}$	72 [min]
Release date of Block i r_i	DU(0, 5) [day]
Due Date Tightness τ	U(0.8, 1.2)
Variability Factor of Processing Time δ	U(0.1, 0.5)

에이전트의 학습이 진행됨에 따른 각 에피소드 별 누적 보상 그래프는 [그림 5-1]와 같다. 약 5,000 에피소드 이후에서 약 -20으로 누적 보상이 수렴한 것을 확인할 수 있다. 즉, set-up과 tardiness를 감소시키는 방향으로 스케줄링 정책이 학습되는 것을 확인할 수 있다.

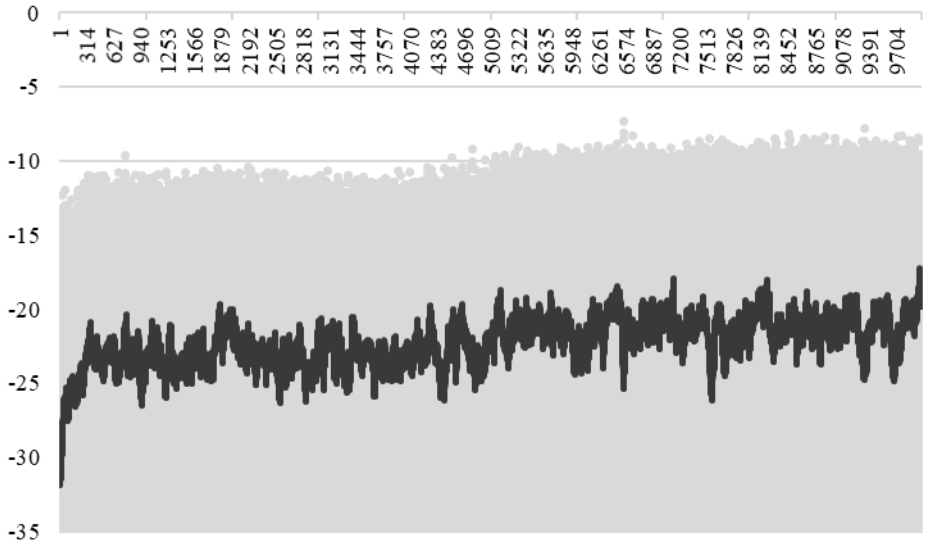


그림 5-1 Graph of Cumulative Reward of PMSP of Shipbuilding

제 6 장 실험

6.1 실험 시나리오

제 5 장에서 학습한 스케줄링 알고리즘의 성능을 평가하기 위하여 Tardiness, Set-up Ratio의 두 가지 지표를 정의하고 테스트 시나리오에 대하여 에이전트의 행동으로 정의한 네 가지 우선순위규칙(SSPT, ATCS, MDD, COVERT rule)과의 비교를 수행하였다.

첫 번째 지표는 블록의 평균 납기 지연$(\overline{T})$이다. 각 블록의 납기 지연을 각 블록의 납기일(d_i)과 해당 블록에 속하는 모든 부재의 작업이 완료된 시점(C_i)의 시간 단위의 차이로 정의한 후, 평균 값을 계산하였다. 즉, $\overline{T}$ 값이 작을수록 목적함수 중 하나인 총 납기지연의 최소화를 효과적으로 달성할 수 있음을 의미한다. 두 번째 지표인 Set-up Ratio(R_σ)는 전체 보강재 중 부재의 작업 시 용접 라인에서의 Set-up 변경이 발생한 보강재의 비율로 정의한다. 본 연구에서의 목적함수 중 하나가 Set-up 시간 최소화이기 때문에 Set-up Ratio가 작을수록 스케줄링 알고리즘의 성능이 좋다는 것을 의미한다.

테스트 시나리오는 동일한 계획 대상 기간을 기준으로 블록의 수를 달리하여 구성하였다. 일주일 기준 물량인 블록 80개에 대하여 계획을 수립할 때와 기준 대비 적은 물량인 60개의 블록에 대하여 투입 순서 계획을 수립할 때, 그리고 기준 대비 많은 물량인 100개의 블록에 대하여 투입 순서 계획을 수립할 때를 비교한다. 또한 작업 시간의 변동성 지수인 δ_{pt}를 0.1, 0.2, 0.3, 0.4, 0.5로 달리하여 실험하였다.

결과적으로 총 15개의 테스트 케이스([표 6-1])에 대하여 강화학습으로 학습한 스케줄링 알고리즘과 우선순위 규칙 중 하나의 규칙만 적용하는 알고리즘의 성능 비교를 수행하였다. 각 모델 간의 성능 비교는 테스트를 총 50번 반복하여 계산한 지표의 평균값에 기반한다.

표 6-1 Description of Test Cases for PMSP of Shipbuilding

	Cases	Number of Cases
Number of Blocks N	60, 800, 100	3
Variability Factor δ_{pt}	0.1, 0.2, 0.3, 0.4, 0.5	5
Total Test Case		15

마지막으로, 실험에서 사용한 블록의 개수에 따른 최적의 ATCS와 COVERT rule의 파라미터는 [표 6-2]와 같다.

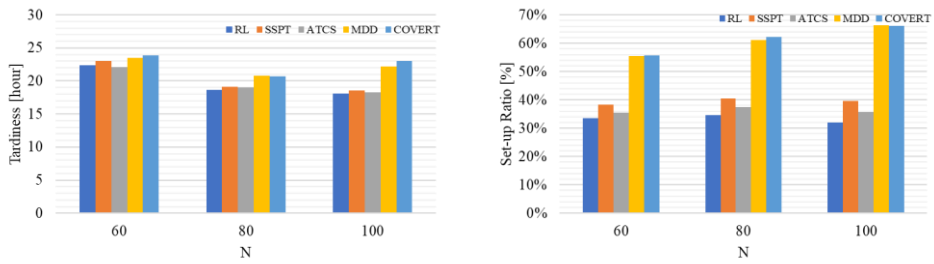
표 6-2 Parameters of Dispatching rules depending on N for PMSP of Shipbuilding

N	ATCS		COVERT
	k_1	k_2	k
60	5.535	0.943	0.4
80	5.953	1.000	6.6
100	6.122	0.953	9.0

6.2 실험 결과

본 절에서는 [표 6-1]에 표현된 15개의 테스트 케이스에 대한 결과를 블록의 개수 (N), 작업 시간의 변동성 지수 (δ_{pt}) 에 따라 정리하였다.

[표 6-3]에서는 블록의 개수 (N) 에 따른 각 지표에 대한 결과를 강화학습의 결과 대비 다른 우선순위 규칙의 결과에 대한 비율과 함께 정리하였다. [그림 6-1]은 [표 6-3]을 그래프로 나타낸 결과이다. 실험 결과, 기준 계획 대비 적은 수의 블록이 들어온 $N = 60$ 에서 ATCS rule을 적용한 결과를 제외하고는 모든 경우에서 본 연구에서 제안한 스케줄링 모델이 평균 Tardiness와 Set-up Ratio가 가장 작은 값을 나타냄을 확인할 수 있다. $N = 60$ 인 경우의 ATCS rule의 Tardiness 지표는 강화학습 대비 약 1.4% 낮은 결과를 보였지만 Set-up Ratio의 경우 강화학습이 약 6% 낮은, 즉 더 좋은 결과를 가진다. 따라서 동일 기간에 서로 다른 개수의 블록이 투입되더라도 본 연구에서 제안한 모델이 Tardiness 최소화와 Set-up 시간 최소화라는 두 가지 목적함수를 동시에 달성하는데 있어 효과적임을 확인할 수 있었다.



(a) Tardiness (b) Set-up Ratio

그림 6-1 Comparison Results depending on N

표 6-3 Performance between Reinforcement Learning and Other Heuristic rules depending on N

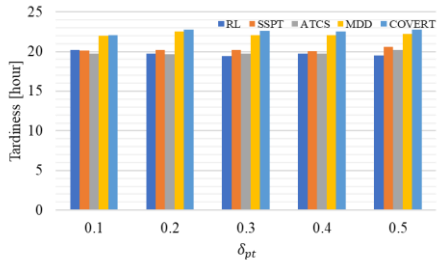
N		RL	SSPT	ATCS	MDD	COVERT
60	$\overline{T}$ [hour]	22.39 (1)	23 (1.027)	22.09 (0.986)	23.53 (1.051)	23.84 (1.065)
	R_σ [%]	33.5 (1)	38.3 (1.143)	35.5 (1.06)	55.5 (1.655)	55.6 (1.661)
80	$\overline{T}$ [hour]	18.66 (1)	19.11 (1.024)	19.04 (1.02)	20.84 (1.116)	20.73 (1.111)
	R_σ [%]	34.6 (1)	40.5 (1.17)	37.5 (1.084)	61.2 (1.769)	62.1 (1.796)
100	$\overline{T}$ [hour]	18.11 (1)	18.56 (1.025)	18.31 (1.011)	22.17 (1.224)	23.04 (1.272)
	R_σ [%]	31.9 (1)	39.6 (1.241)	35.7 (1.12)	66.2 (2.076)	66.2 (2.074)

[표 6-4]에서는 작업시간의 변동성 지수(δ_{pt})에 따른 각 지표에 대한 결과를 강화학습의 결과 대비 다른 우선순위 규칙의 결과에 대한 비율과 함께 정리하였다. [그림 6-2]은 [표 6-4]을 그래프로 나타낸 결과이다. 실험 결과, 낮은 변동성 $(\delta_{pt} = 0.1, 0.2)$에서는 강화학습으로 학습한 스케줄링 모델이 일부 우선순위 규칙을 적용한 경우보다 더 큰 납기지연을 보였지만, 변동성이 커질수록 가장 낮은 납기지연을 보였다. 또한 변동성 지수에 따른 Tardiness 지표 변화 그래프([그림 6-3])을 확인했을 때, 다른 우선순위 규칙의 경우 변동성이 커질수록 블록의 평균 납기 지연이 심해지는 것을 확인할 수 있다. 반면 본 연구의 스케줄링 모델은 작업시간의 변동성이 커져도 납기 지연에 대한 지표가 좋아지는 것을 확인할 수 있다. Set-up Ratio의 경우 본 연구의 스케줄링 모델이 가장 낮은 Set-up Ratio을 보였다. 즉, 작업 시간의 변동성이 달라지는 환경에 있어 본 연구에서 제안한 모델이 Tardiness

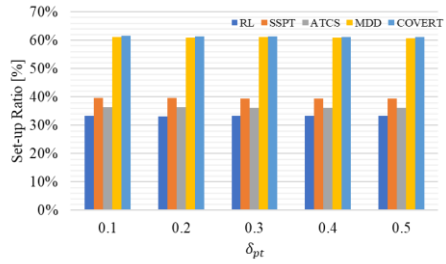
최소화와 Set-up 시간 최소화라는 두 가지 목적함수를 동시에 달성하는데 있어 효과적임을 확인할 수 있었다.

표 6-4 Performance between Reinforcement Learning and Other Heuristic rules depending on δ_{pt}

δ_{pt}		RL	SSPT	ATCS	MDD	COVERT
0.1	$\overline{T}$ [hour]	20.18 (1)	20.09 (0.995)	19.72 (0.977)	22.01 (1.090)	22.06 (1.093)
	R_σ [%]	33.44 (1)	39.5 (1.183)	36.3 (1.086)	61.2 (1.830)	61.6 (1.841)
0.2	$\overline{T}$ [hour]	19.75 (1)	20.20 (1.023)	19.65 (0.995)	22.50 (1.140)	22.73 (1.151)
	R_σ [%]	33.2 (1)	39.5 (1.191)	36.3 (1.093)	60.8 (1.833)	61.4 (1.849)
0.3	$\overline{T}$ [hour]	19.44 (1)	20.16 (1.037)	19.71 (1.014)	22.08 (1.136)	22.62 (1.164)
	R_σ [%]	33.3 (1)	39.5 (1.186)	36.3 (1.089)	61.1 (1.837)	61.3 (1.843)
0.4	$\overline{T}$ [hour]	19.71 (1)	20.05 (1.017)	19.76 (1.002)	22.05 (1.119)	22.55 (1.144)
	R_σ [%]	33.4 (1)	39.4 (1.179)	36.2 (1.084)	60.9 (1.824)	61.2 (1.833)
0.5	$\overline{T}$ [hour]	19.53 (1)	20.62 (1.056)	20.21 (1.035)	22.24 (1.139)	22.72 (1.164)
	R_σ [%]	33.4 (1)	39.3 (1.178)	36.2 (1.084)	60.7 (1.820)	61.1 (1.831)



(c) Tardiness



(d) Set-up Ratio

그림 6-2 Comparison Results depending on δ_{pt}

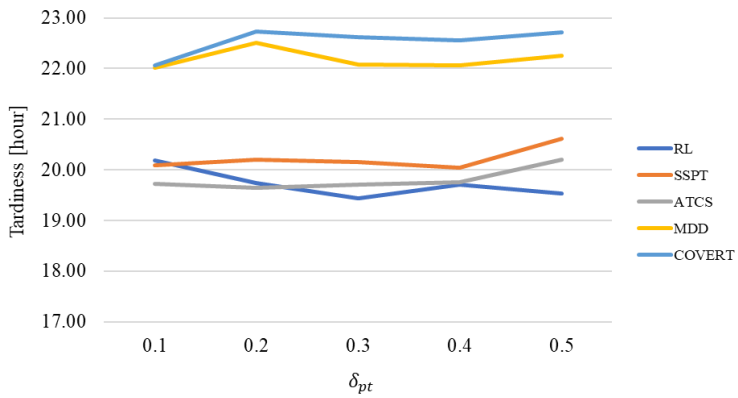


그림 6-3 Trends on Tardiness depending on δ_{pt}

제 7 장 일반 문제 확장

제 6 장까지의 학습과 실험을 통해 조선소의 존재하는 병렬 기계 스케줄링 문제 중 기계 단위의 문제인 형강공장의 용접 라인 스케줄링 문제에 본 논문에서 제안한 MDP와 학습 알고리즘이 효과적으로 적용됨을 확인하였다. 그러나 본 논문에서 제안한 스케줄링 알고리즘은 조선소 형강공장에 존재하는 병렬 용접 기계에 특화되어 개발된 알고리즘으로, 조선소에 존재하는 다른 병렬 기계 스케줄링 문제에도 효과적으로 응용되는 지 확인하기 위해서는 일반화된 문제에 대한 테스트가 필수적이다. 따라서 제 4 장과 제 5 장에서 제안한 MDP와 학습 알고리즘을 문제 세팅을 병렬 기계 스케줄링 문제를 일반화된 형태로 변경한 개념적인 문제에 적용하여 조선소에 존재하는 다른 병렬 기계 문제로의 확장성에 대한 타당성 검토를 진행한다.

7.1 문제 정의

7.1.1 문제 가정

실제 조선소의 생산 데이터를 바탕으로 문제를 구성한 제 3 장과 달리 본 장에서는 분포함수를 통해 데이터를 생성한다. 일반문제는 n개의 Job, m개의 Machine으로 이루어져 있으며, Job 별로 평균 작업 시간$(\overline{p_i})$, 특성(θ_i) 그리고 납기일(d_i)에 대한 정보를 가진다. 기계의 경우 형강공정 스케줄링 문제와 마찬가지로 set-up status θ_k를 가지며, 실제 작업 시간은 변동성 지수 δ에 따라 결정된다. 일반 문제의 거동을 모사한 시뮬레이션 흐름은 [그림 7-1]과 같다. 또한 형강공장의 병렬 용접기계 문제(제 3 장)와 일반 문제(제 7 장)에서의 문제 가정의 차이점은 [표 7-1]로 정리하였다.

표 7-1 Difference between Profile Shop Problem and General Problem

	Profile Shop Problem	General Problem
Object	Job(Steel), Job Family(Block)	Job
Set-up Time	Fixed (5 min)	Non-Fixed
Processing Time	Average processing time : Given	Average processing time : uniform distribution

스케줄링의 대상 또는 시뮬레이션의 객체의 관점에서, 형강공장의 스케줄링 문제는 Job(부재)과 Job Family(블록)의 개념으로 구성되어 실제 작업의 객체는 Job(부재)이며, 납기일 (d_i) 의 속성은 Job Family(블록)에 따라 결정된다. 반면 일반 문제는 Job - Job Family의

구성이 아닌 단일한 Job의 구성으로만 이루어져 있으며, 납기일 등의 속성 또한 Job에 따라 결정된다. Set-up 시간 관점에서, 형강공정의 스케줄링 문제는 용접라인 M_k의 셋업 값(θ_k)과 작업 대상 부재 S_{ij}의 특성 (θ_{ij})이 다르면 5분의 고정된 Set-up 시간이 발생했다. 반면 일반문제에서는 기계 M_k의 셋업 값(θ_k)과 작업 대상 J_i의 특성(θ_i)가 다르면 식 (40)을 통해 계산된 값만큼 Set-up 시간이 발생한다고 가정하였다. 작업 시간 관점에서는, 형강공장 스케줄링 문제에서는 부재 S_{ij} 의 평균 작업 시간 $(\overline{p}_{ij})$ 는 주어진 데이터를 기반으로 계산된 값이었다면, 일반 문제에서는 Job J_i의 평균 작업 시간$(\overline{p}_i)$는 균등분포를 통해 주어진다.

일반 문제 스케줄링 알고리즘의 결정 변수는 각 Job의 작업 시작 시간과 작업이 이루어질 기계이다. 이를 통해 달성하고자 하는 목적함수는 납기 지연 시간의 최소화와 Job이 기계에서 작업할 때 발생하는 셋업 시간의 최소화이다. Job 별로 발생하는 납기 지연은 식 (41)을 통해 계산되며, 두 목적함수는 식 (42), (43)을 통해 나타낼 수 있다.

$$\sigma_{ik} = |\theta_i - \theta_k| \tag{40}$$

$$T_i\,[day] = max(0, C_i - d_i) \tag{41}$$

$$Minimize \sum_{i=1}^{N} T_i \tag{42}$$

$$Minimize \sum_{i=1}^{n} \sigma_{ik} \tag{43}$$

그림 7-1 Simulation Framework of General Problem

7.1.2 Markov Decision Process

Markov Decision Process를 이루는 상태, 행동, 보상은 제 4 장에서 제안된 구성과 거의 동일하게 구성했다. 상태는 4.1에서 제안한 특성벡터들로 이루어져있다. set-up 최소화와 관련된 f_1 (식 (44)), Tardiness 및 set-up 최소화와 관련하여 Tardiness level로 계산되는 f_2, f_3 (식 (45)에서 (47)), 그리고 각 기계의 현재 작업 중인 Job의 진도율인 f_4(식 (48))로 구성되어 있으며, 상태의 크기는 $2m+8$이다.

$$f_{1,k} = \begin{cases} \dfrac{N_k}{N_q} & If\ Machine\ M_k's\ set-up\ is\ detemined \\ 1.0 & Otherwise\ (Initial\ State) \end{cases}, \quad (k = 1, \dots, m) \tag{44}$$

N_k : the number of jobs that satisfy condition $\theta_k = \theta_i$

N_q : the number of jobs in $Q(t)$

$$f_{2(3),\,g} = \frac{N_{2(3),g}}{N_{w,\,2(3)}} \quad (g = 1,\ 2,\ 3,\ 4)$$

$N_{2(3),g}$: the number of jobs which has tardiness level g

$N_{q,\,2(3)}$: the number of jobs which have not been worked yet that satisfy condition $\theta_k = \theta_i (f_2)$ or $\theta_k \neq \theta_i\ (f_3)$ (45)

$$Tardiness\ Level\ g = \begin{cases} 1, & if\ d_i - t \in (max\, r_i\ ,\ +\infty) \\ 2, & if\ d_i - t \in (min\, r_i\ , max\, r_i] \\ 3, & if\ d_i - t \in (0, min\, r_i] \\ 4, & if\ d_i - t \in (-\infty,\ 0] \end{cases} \tag{46}$$

$$max\, r_i = \left(1 + \delta_{pt}\right) \times \overline{p}_i$$
$$min\, r_i = \left(1 - \delta_{pt}\right) \times \overline{p}_i \tag{47}$$

$$f_{4,k} = \begin{cases} \dfrac{t_{s,i,k} + \overline{p}_i - t}{\overline{p}_i}, & if\ \ Machine\ M_k\ is\ working \\ 0, & if\ \ Machine\ M_k\ is\ idle \end{cases}, \quad (k = 1, \dots, m) \tag{48}$$

$t_{s,i,k}$: *the start time of Job* J_i *in Machine* M_k

행동 또한 4.2와 동일하게 의사결정 시점 t에서 작업 가능한 Job의 집합인 $Q(t)$ 중 다음으로 작업할 Job을 선택하기 위한 우선순위 규칙을 선택하는 것으로 정의했으며, SSPT, ATCS, MDD, COVERT rule을 사용하였다. 각 우선순위 규칙에 따라 의사결정 시점 t에서의 Machine M_k를 위한 Job J_i는 식 (49)에서 (52)을 통해 결정된다([표 7-2]). ATCS rule의 파라미터인 k_1과 k_2는 표 4-2에 따라 결정하였고, COVERT rule의 파라미터인 k는 [0.1, 20]의 범위에서 case study를 하여 가장 작은 mean tardiness를 가지는 k로 결정하였다.

표 7-2 Formulation of Each Heuristic rule

Heuristic	Formulation
SSPT	$J_i = argmin\{\sigma_{ik} + \overline{p}_i, \forall J_i \in Q(t)\}$ (49)
ATCS	$J_i = argmax\left\{\begin{array}{c}\frac{1}{\bar{p}_i} exp\left(-\frac{max(d_i - \bar{p}_i - t, 0)}{k_1 \bar{p}}\right) exp\left(-\frac{\sigma_{ik}}{k_2 \overline{\sigma}}\right), \\ \forall J_i \in Q(t)\end{array}\right\}$ (50) $\overline{p} = \frac{1}{N_q}\sum^{N_q} \overline{p}_i \ , \quad \overline{\sigma} = \frac{1}{N_q}\sum^{N_q} \sigma_{ik}$
MDD	$J_i = argmin\{\max(d_i, t + \overline{p}_i), \forall J_i \in Q(t)\}$ (51)
COVERT	$J_i = argmax\left\{\begin{array}{c}\frac{1}{\bar{p}_i} max\left(1 - \frac{max(d_i - \bar{p}_i - t, 0)}{k\bar{p}_i}, 0\right), \\ \forall J_i \in Q(t)\end{array}\right\}$ (52)

보상 또한 4.3과 거의 유사하게 정의하였다. Tardiness 최소화와 관련된 보상 r_T는 의사결정 시점 사이에 작업을 마친 Job의

Tardiness를 식 (53)와 같이 [−1, 0] 사이로 변환한 각 Job의 penalty의 합으로 정의하였다(식 (54)).

Set−up 시간 최소화 관련된 보상 r_S는 식 (55)과 같이 정의된다. r_T 와 마찬가지로 r_S 를 [−1, 0] 범위로 변환하였다. 구체적으로, 에이전트가 선택한 우선순위 규칙에 따라 Machine M_k 에서 다음으로 작업할 Job J_i를 선택하여 작업하였을 때 set−up 시간 σ_{ik}가 발생하면 발생한 σ_{ik}을 발생 가능한 set−up 시간의 최대값으로 나누어주었다.

의사결정 시점 t에서의 최종 보상 r_t는 식 (56)과 같이 두 보상의 가중합으로 계산된다.

$$r_{T,i} = e^{\max(C_i - d_i, 0)} - 1 \tag{53}$$

$$r_T = \sum r_{T,i} \tag{54}$$

$$r_S = \frac{\sigma_{ik}}{max\ \sigma_{ik}} \tag{55}$$

$$r_t = 0.6 \times r_T + 0.4 \times r_S \tag{56}$$

7.2 학습 및 실험

7.2.1 학습

확률 분포로 구성된 일반적인 병렬 기계 스케줄링 문제에 대하여 7.1.2에서 제안한 MDP와 PPO 알고리즘을 기반으로 학습을 진행하였다. 학습에 사용한 인공신경망은 제 5 장과 동일하게 총 5개의 완전 연결계층으로 구성되어 있다. 입력층은 상태 벡터의 크기와 동일한 $2m+8$ 개의 노드를 갖고, 세 개의 은닉층은 차례로 512개, 512개, 256개의 노드를 갖는다. 마지막으로 출력층은 전체 행동의 수와 동일한 4개의 노드를 갖는다. 이 때 마지막 출력층을 제외한 첫 번째에서 네 번째 층은 모두 활성함수로 ReLU 함수를 사용한다.

일반문제의 학습을 위한 PPO 알고리즘의 하이퍼파라미터의 설정을 위하여 제 5 장과 동일하게 Learning Rate α를 [0.005, 0.001, 0.0005, 0.0001] 범위에서, Optimization Epoch K를 1또는 2로 달리한 총 8개의 하이퍼파라미터 셋에 대해 실험을 진행하였고, 가장 높은 성능을 보인 조합을 사용하였다([표 7-3]). 에이전트는 총 5,000번동안 학습을 수행하였다.

PPO 알고리즘을 이용한 학습의 pseudocode와 Hyper-parameter는 Learning Rate α를 제외하고 [표 5-1], [표 5-2]와 동일하다. 최적화 알고리즘은 Learning Rate α를 0.001로 설정한 Adam 알고리즘을 적용한다.

표 7-3 Hyper-Parameters in the Learning Phase for General Problem

Hyper-parameter	Value
Number of Episodes E	10,000
Optimization Epoch K	1
Clipping Parameter ε	0.2
Discount rate γ	0.98
GAE Parameter λ	0.95
Learning Rate α	0.001

시뮬레이션에 사용한 분포함수와 파라미터는 [표 7-4]와 같다. 100개의 Job을 5개의 기계에서 처리하는 문제를 가정했으며, 각 Job에 할당된 평균 작업 시간은 최소값을 10, 최대값을 20으로 갖는 균등분포에 의해 결정하였다. 따라서 IAT의 평균으로는 각 기계의 평균 작업시간을 기계 대수로 나눈 3의 값을 사용하였다. 또한 Job이 기계에서 작업할 때의 셋업 시간을 결정짓는 Job property인 θ_i는 0에서 5사이의 정수값을 할당하였다. 따라서 식 (55)에서의 가능한 셋업 시간은 최대값 $(\max \sigma_{ik})$ 은 5가 된다. 마지막으로 due date tightness (τ)와 작업 시간의 변동성 지수 (δ_{pt})는 제 5 장과 동일하게 설정하였다. 네 개의 행동 중 ATCS rule의 파라미터는 [표 4-2]에 의해 $k_1 = 2.73,\ k_2 = 1.153$으로 결정하였다. 마찬가지로 COVERT rule의 파라미터는 case study를 통해 $k = 6.8$으로 결정하였다.

표 7-4 Simulation Parameters in the Learning Phase for General Problem

Parameter	Value
The Number of Jobs n	100
The Number of Machines m	5
Average Processing Time $\overline{p}_i$	U(10, 20)
Average of IAT $\frac{1}{\lambda}$	3 $\left(=\frac{\overline{p}}{m}=\frac{15}{5}\right)$
Job Property θ_i	DU(0, 5)
Due Date Tightness τ	U(0.8, 1.2)
Variability Factor of Processing Time δ_{pt}	U(0.1, 0.5)

에이전트의 학습에 따른 각 에피소드 별 누적보상 그래프는 [그림 7-2]와 같다. 학습 초기의 누적보상 -65에서 약 400 에피소드 이후에 약 -55로 증가하여 수렴하는 것을 볼 수 있다. 즉, set-up과 tardiness를 감소시키는 방향으로 스케줄링 정책이 학습되는 것을 확인할 수 있다.

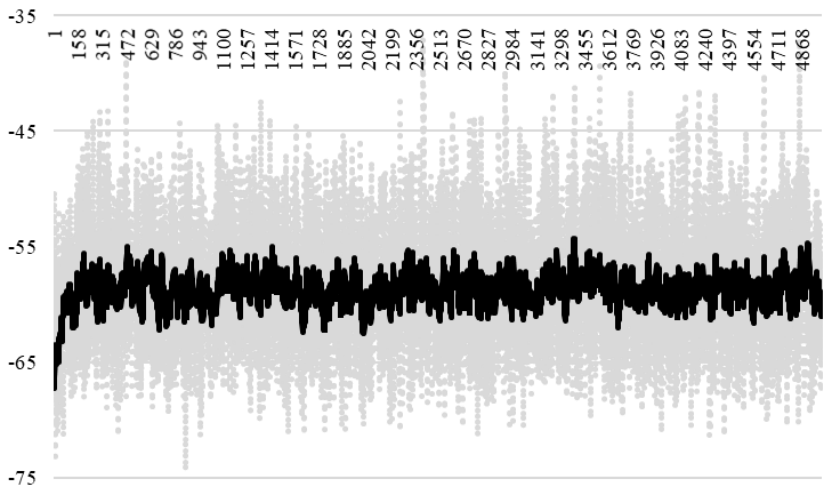


그림 7-2 Graph of Cumulative Reward of General Problem Setting

7.2.2 실험

7.2.1에서 학습된 스케줄링 알고리즘의 성능을 평가하기 위하여 Tardiness, Set-up time의 두 가지 지표를 정의하고 테스트 시나리오에 대하여 에이전트의 행동으로 정의한 네 가지 우선순위 규칙과의 비교를 수행하였다.

첫 번째 지표인 Tardiness는 6.1과 동일하게 각 Job의 d_i 대비 C_i의 차이로 정의하고 Job의 평균 Tardiness$(\overline{T})$를 계산하였다. 두 번째 지표인 Set-up time의 경우 각 Job의 평균 set-up time $(\overline{\sigma})$ 을 계산하였는데, 이는 형강공정의 스케줄링 문제와 달리 set-up 시간이 고정되지 않고 0에서 5사이의 값을 가지기 때문에 set-up이 발생한 Job 개수의 비율이 아닌 시간 그 자체의 평균을 평가 지표로 정의하였다.

테스트 시나리오는 계획 대상 Job의 개수(n)를 달리한다. 이 때 동일한 조건에서 성능을 평가하기 위하여 due date tightness, 그리고 작업 시간의 변동성 지수인 δ_{pt}를 고정하여 실험을 진행하였다. [표 7-5]에서는 테스트 케이스를 정리하였다. 각 모델간 성능 비교는 테스트를 총 100번 반복하여 계산한 각 지표의 평균값에 기반한다. [표 7-6]에서는 Job의 개수에 따른 ATCS와 COVERT rule의 파라미터를 정리하였다.

표 7-5 Description of Test Cases for General Problem Setting

	Cases	Number of Cases
Number of Jobs n	100, 200, 400	3
Due Date Tightness τ	0.8, 1.0, 1.2	3
Variability Factor δ_{pt}	0.1, 0.2, 0.3, 0.4, 0.5	5
Total Test Case		45

표 7-6 Parameters of Dispatching rules depending on n for General Problem Setting

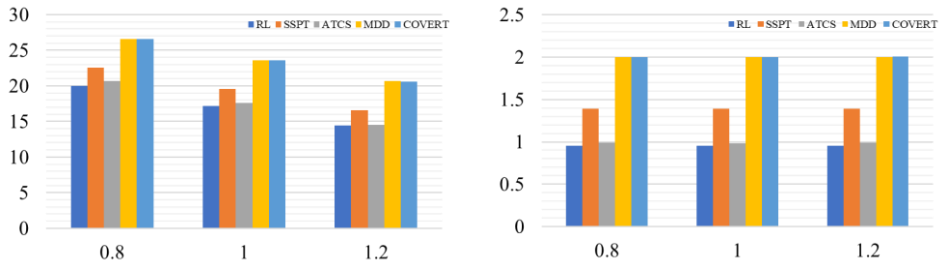
n	ATCS		COVERT
	k_1	k_2	k
100	2.730	1.153	6.8
200	3.519	1.252	4.4
400	3.338	1.209	3.9

7.2.3 결과

본 절에서는 [표 7-5]의 45개의 테스트 케이스에 대한 결과를 due date tightness (τ), 작업 시간의 변동성 지수 (δ_{pt}) 에 따라 정리하였다.

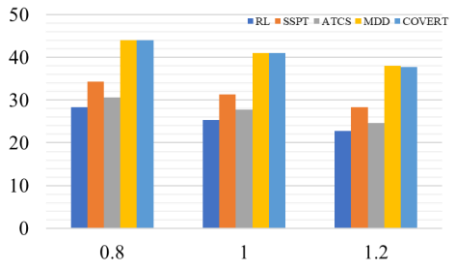
[그림 7-3]에서는 due date tightness(τ)에 따른 두 지표의 결과를 Job의 개수(n)에 따라 정리하였다. 전체적으로 due date tightness의 값이 커짐에 따라 평균 Tardiness의 값이 작아지는 경향을 확인할 수 있다. 하지만 due date tightness는 납기일 설정에만 관여하기 때문에 due date tightness의 값이 변화해도 Set-up time의 값의 변동이 발생하지 않았다.

모든 경우에서 본 연구에서 제안한 스케줄링 모델이 다른 우선순위 규칙 대비 가장 작은 평균 Tardiness와 Set-up time 값을 가지는 것을 확인할 수 있었다. [표 7-7]에서 [표 7-9]는 각 모델의 결과와 스케줄링 모델 대비 다른 우선순위 규칙의 결과의 비율을 병기하여 정리하였다. Tardiness의 경우 최소 0.4%에서 최대 112%의 차이가 발생했고, Set-up time의 경우 최소 4%에서 최대 245%의 차이가 발생했다. 즉, 본 연구에서 제안한 스케줄링 모델이 due date tightness가 변화하는 환경에서 Tardiness 최소화와 Set-up time 최소화라는 두 가지 목적함수를 동시에 만족하는 데 있어 효과적임을 확인할 수 있었다.

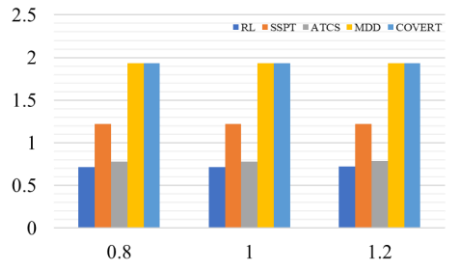


(a) n = 100, Tardiness

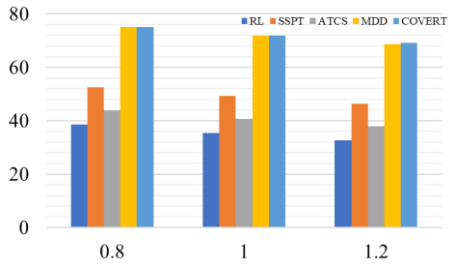
(b) n = 100, Set-up Time



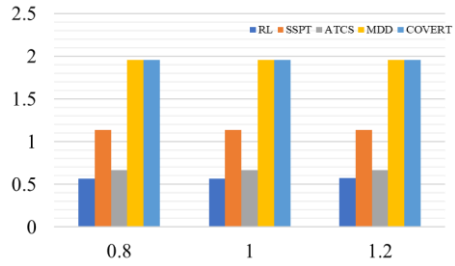
(c) n = 200, Tardiness



(d) n = 200, Set-up Time



(e) n = 400, Tardiness



(f) n = 400, Set-up Time

그림 7-3 Comparison Results depending on τ, n

표 7-7 Performance between Reinforcement Learning and Other Heuristic rules depending on τ under $n = 100$

$n = 100$						
τ		RL	SSPT	ATCS	MDD	COVERT
0.8	$\overline{T}$	19.99 (1)	22.56 (1.127)	20.68 (1.035)	26.59 (1.330)	26.59 (1.330)
	$\overline{\sigma}$	0.951 (1)	1.394 (1.466)	0.986 (1.037)	2.002 (2.106)	2.002 (2.106)
1.0	$\overline{T}$	17.13 (1)	19.54 (1.140)	17.59 (1.026)	23.59 (1.377)	23.59 (1.377)
	$\overline{\sigma}$	0.951 (1)	1.394 (1.466)	0.986 (1.036)	2.002 (2.106)	2.002 (2.106)
1.2	$\overline{T}$	14.46 (1)	16.54 (1.144)	14.51 (1.004)	20.70 (1.431)	20.56 (1.422)
	$\overline{\sigma}$	0.954 (1)	1.394 (1.461)	0.988 (1.036)	2.002 (2.099)	2.007 (2.104)

표 7-8 Performance between Reinforcement Learning and Other Heuristic rules depending on τ under $n = 200$

$n = 200$						
τ		RL	SSPT	ATCS	MDD	COVERT
0.8	$\overline{T}$	28.31 (1)	34.33 (1.213)	30.66 (1.083)	43.97 (1.553)	43.97 (1.553)
	$\overline{\sigma}$	0.716 (1)	1.222 (1.706)	0.781 (1.091)	1.932 (2.698)	1.932 (2.698)
1.0	$\overline{T}$	25.41 (1)	31.31 (1.232)	27.80 (1.094)	40.95 (1.612)	40.95 (1.612)
	$\overline{\sigma}$	0.716 (1)	1.222 (1.706)	0.779 (1.088)	1.932 (2.698)	1.932 (2.698)
1.2	$\overline{T}$	22.78 (1)	28.30 (1.242)	24.62 (1.081)	37.98 (1.667)	37.76 (1.658)
	$\overline{\sigma}$	0.719 (1)	1.222 (1.699	0.786 (1.093)	1.932 (2.686)	1.932 (2.686)

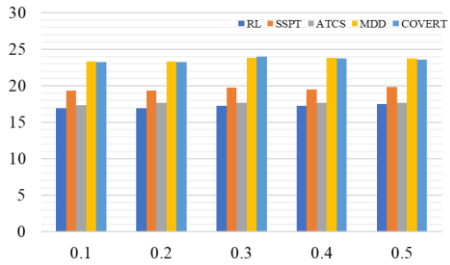
표 7-9 Performance between Reinforcement Learning and Other Heuristic rules depending on τ under $n = 400$

$n = 400$						
τ		RL	SSPT	ATCS	MDD	COVERT
0.8	$\overline{T}$	38.56 (1)	52.41 (1.359)	43.94 (1.140)	75.10 (1.948)	75.10 (1.948)
	$\overline{\sigma}$	0.568 (1)	1.134 (1.996)	0.668 (1.175)	1.957 (3.445)	1.957 (3.445)
1.0	$\overline{T}$	35.50 (1)	49.24 (1.387)	40.58 (1.143)	71.92 (2.026)	71.92 (2.026)
	$\overline{\sigma}$	0.568 (1)	1.134 (1.996)	0.666 (1.173)	1.957 (3.445)	1.957 (3.445)
1.2	$\overline{T}$	32.64 (1)	46.24 (1.417)	38.02 (1.165)	68.69 (2.105)	69.13 (2.118)
	$\overline{\sigma}$	0.571 (1)	1.134 (1.985)	0.665 (1.164)	1.956 (3.424)	1.956 (3.424)

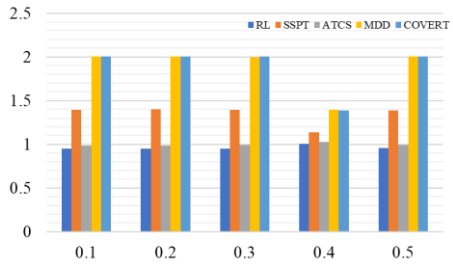
[그림 7-4]에서는 작업 시간의 변동성 지수(δ_{pt})에 따른 두 지표의 결과를 Job의 개수 (n) 에 따라 정리하였다. 전체적으로 작업시간의 변동성 지수가 커짐에 따라 평균 Tardiness의 값이 커지는 경향을 확인할 수 있다. 반면, due date tightness를 달리 했을 때와 마찬가지로 작업시간의 변동성 지수를 달리해도 Set-up time의 값에는 변동이 거의 존재하지 않았다.

모든 경우에서 본 연구에서 제안한 스케줄링 모델이 다른 우선순위 규칙 대비 가장 작은 평균 Tardiness와 Set-up time값을 가지는 것을 확인할 수 있었다. [표 7-10], [표 7-11] 그리고 [표 7-12]에서는 각 모델의 결과와 스케줄링 모델 대비 다른 우선순위 규칙의 결과의 비율을 병기하여 정리하였다. Tardiness의 경우 최소 1.2%에서 최대 106%의 차이가 발생했고, Set-up time의 경우 최소 2.0%에서 최대 249%의 차이가 발생했다. 즉, 본 연구에서 제안한 스케줄링 모델이 작업시간의

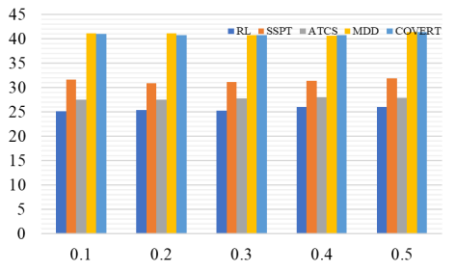
변동성이 변화하는 환경에서 Tardiness 최소화와 Set-up time 최소화라는 두 가지 목적함수를 동시에 만족하는 데 있어 효과적임을 확인할 수 있었다.



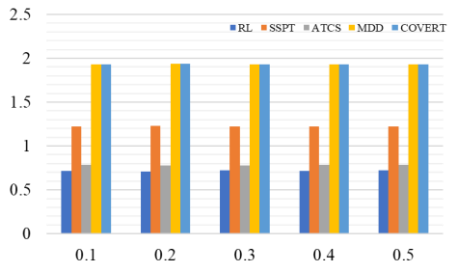
(g) n = 100, Tardiness



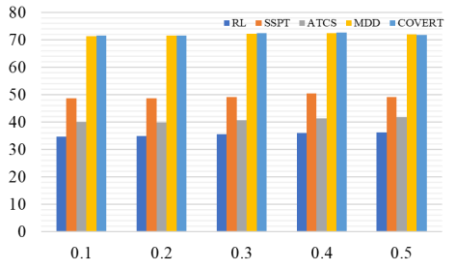
(h) n = 100, Set-up Time



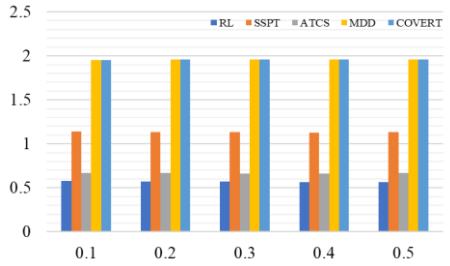
(i) n = 200, Tardiness



(j) n = 200, Set-up Time



(k) n = 400, Tardiness



(l) n = 400, Set-up Time

그림 7-4 Comparison Results depending on δ_{pt}, n

표 7-10 Performance between Reinforcement Learning and Other Heuristic rules depending on δ_{pt} under $n = 100$

$n = 100$						
δ_{pt}		RL	SSPT	ATCS	MDD	COVERT
0.1	$\overline{T}$	16.97 (1)	19.31 (1.138)	17.34 (1.022)	23.37 (1.377)	23.27 (1.371)
	$\overline{\sigma}$	0.948 (1)	1.393 (1.470)	0.987 (1.042)	2.001 (2.112)	2.005 (2.115)
0.2	$\overline{T}$	16.96 (1)	19.34 (1.140)	17.64 (1.040)	23.33 (1.376)	23.26 (1.372)
	$\overline{\sigma}$	0.949 (1)	1.402 (1.477)	0.983 (1.036)	2.004 (2.112)	2.005 (2.113)
0.3	$\overline{T}$	17.30 (1)	19.73 (1.140)	17.65 (1.020)	23.84 (1.378)	24.01 (1.387)
	$\overline{\sigma}$	0.952 (1)	1.394 (1.464)	0.989 (1.039)	2.001 (2.102)	2.004 (2.105)
0.4	$\overline{T}$	17.26 (1)	19.50 (1129)	17.64 (1.022)	23.81 (1.379)	23.74 (1.375)
	$\overline{\sigma}$	1.006 (1)	1.136 (1.130)	1.026 (1.020)	1.391 (1.384)	1.387 (1.379)
0.5	$\overline{T}$	17.48 (1)	19.81 (1.134)	17.70 (1.012)	23.79 (1.361)	23.63 (1.352)
	$\overline{\sigma}$	0.959 (1)	1.390 (1.449)	0.991 (1.034)	2.005 (2.090)	2.004 (2.090)

표 7-11 Performance between Reinforcement Learning and Other Heuristic rules depending on δ_{pt} under $n = 200$

$n = 200$						
δ_{pt}		RL	SSPT	ATCS	MDD	COVERT
0.1	$\overline{T}$	25.10 (1)	31.53 (1.256)	27.40 (1.092)	41.12 (1.638)	40.99 (1.633)
	$\overline{\sigma}$	0.714 (1)	1.219 (1.706)	0.782 (1.095)	1.931 (2.702)	1.931 (2.702)
0.2	$\overline{T}$	25.35 (1)	30.88 (1.218)	27.51 (1.085)	41.07 (1.620)	40.68 (1.605)
	$\overline{\sigma}$	0.711 (1)	1.227 (1.725)	0.781 (1.099)	1.935 (2.722)	1.936 (2.723)
0.3	$\overline{T}$	25.18 (1)	31.07 (1.234)	27.73 (1.102)	40.76 (1.619)	40.74 (1.618)
	$\overline{\sigma}$	0.720 (1)	1.221 (1.696)	0.779 (1.081)	1.931 (2.681)	1.930 (2.681)
0.4	$\overline{T}$	25.92 (1)	31.30 (1.208)	27.98 (1.079)	40.59 (1.566)	40.71 (1.571)
	$\overline{\sigma}$	0.717 (1)	1.220 (1.701)	0.784 (1.093)	1.932 (2.693)	1.931 (2.692)
0.5	$\overline{T}$	25.96 (1)	31.79 (1.225)	27.85 (1.073)	41.29 (1.590)	41.34 (1.593)
	$\overline{\sigma}$	0.723 (1)	1.222 (1.691)	0.784 (1.085)	1.930 (2.671)	1.931 (2.672)

표 7-12 Performance between Reinforcement Learning and Other Heuristic rules depending on δ_{pt} under $n = 400$

$n = 200$						
δ_{pt}		RL	SSPT	ATCS	MDD	COVERT
0.1	$\overline{T}$	34.72 (1)	48.59 (1.400)	39.93 (1.150)	71.28 (2.053)	71.54 (2.061)
	$\overline{\sigma}$	0.576 (1)	1.140 (1.978)	0.668 (1.159)	1.955 (3.391)	1.955 (3.392)
0.2	$\overline{T}$	34.96 (1)	48.76 (1.395)	39.79 (1.138)	71.48 (2.045)	71.57 (2.048)
	$\overline{\sigma}$	0.571 (1)	1.134 (1.985)	0.667 (1.168)	1.957 (3.426)	1.957 (3.426)
0.3	$\overline{T}$	35.57 (1)	49.12 (1.381)	40.74 (1.145)	72.20 (2.030)	72.40 (2.035)
	$\overline{\sigma}$	0.570 (1)	1.131 (1.984)	0.663 (1.164)	1.958 (3.433)	1.958 (3.434)
0.4	$\overline{T}$	36.06 (1)	50.55 (1.402)	41.41 (1.148)	72.36 (2.006)	72.62 (2.014)
	$\overline{\sigma}$	0.561 (1)	1.128 (2.011)	0.664 (1.183)	1.957 (3.487)	1.958 (3.488)
0.5	$\overline{T}$	36.28 (1)	49.19 (1.356)	41.86 (1.154)	71.92 (1.982)	71.82 (1.980)
	$\overline{\sigma}$	0.566 (1)	1.135 (2.004)	0.668 (1.180)	1.956 (3.454)	1.956 (3.453)

두 결과를 종합해보면 실제 생산환경에 존재할 수 있는 변동성인 due date tightness와 작업시간의 변동성 지수가 변화하는 환경에 있어, 본 연구에서 제안한 스케줄링 모델이 Tardiness와 Set-up time을 줄이는 데 효과적인 모델임을 확인할 수 있었다.

제 8 장 결론

본 논문에서는 조선소 형강공장에서의 생산성 향상을 달성하기 위한 방안으로서, 작업 투입 시간 간격과 작업 시간의 변동성이 존재하는 형강공장 내 병렬 용접 라인에 대하여 보강재의 작업 순서를 결정하는 강화학습 기반의 동적 스케줄링 알고리즘을 개발하였다. 생산 공정에서의 생산성 향상은 구체적으로 납기 지연 최소화와 셋업 시간 최소화라는 구체적인 목적함수로 정의하였다. 그리고 오픈소스 시뮬레이션 도구인 SimPy를 이용하여 시뮬레이션 기반 강화학습 모델을 구축하고 두 가지 목적함수를 달성할 수 있는 스케줄링 정책을 학습하였다. 이 때 학습을 위하여 형강공정 작업 순서 결정 문제에 대해 두 목적함수를 모두 고려할 수 있는 MDP 모델과 학습 알고리즘을 제안하였다.

개발된 알고리즘은 동일 기간 내 투입 물량이 다른 환경에 대하여 행동으로 정의됐던 우선순위 규칙들과의 비교를 통해 성능을 평가하였다. 결과적으로 블록 별 평균 납기지연과 셋업 비율이라는 두 가지 지표를 모두 고려하였을 때, 에이전트가 학습한 스케줄링 정책이 단일한 우선순위 규칙만 적용하는 경우보다 두 가지 목적함수를 동시에 달성하는 데 효과적임을 확인할 수 있었다.

추가적으로 본 연구에서 제안한 MDP와 학습 알고리즘이 조선소에 존재하는 다른 병렬 기계 문제에도 효과적인지 확인하기 위하여 분포함수로 구성한 병렬 기계 스케줄링 문제에 대하여 타당성 검토를 진행하였다. 그 결과 분포함수로 구성된 일반 문제에 대해서도 본 연구에서 제안한 MDP와 학습 알고리즘으로 학습한 스케줄링 정책이

다른 우선순위규칙만을 적용한 경우보다 평균 Tardiness, 평균 Set-up time로 정의한 두 가지 지표 모두에서 가장 좋은 결과를 나타냈다. 따라서 일반적인 PMSP 문제에 대하여 본 연구의 스케줄링 알고리즘이 Tardiness 최소화와 Set-up 시간 최소화로 정의되는 두 가지 목적함수를 동시에 달성하기에 유효한 수단이라는 것을 확인할 수 있었다.

다만, 본 연구에서는 Tardiness 최소화를 위한 보상과 Set-up 시간 최소화를 위한 보상 사이의 가중치를 하나의 값으로 고정하고 학습을 수행하였고, 최적의 가중치를 찾기 위하여 많은 계산시간이 소요되었다. 향후에는 두 목적함수의 다양한 가중치 조합에 대하여 최적의 의사결정을 내릴 수 있도록 스케줄링 알고리즘을 개선하는 방향으로 연구를 진행할 계획이다.

참고 문헌

Lee, J., Kim, H., 1995, Erection process planning & scheduling using genetic algorithm, *Journal of the Society of Naval Architects of Korea*, 32(1), pp. 9-16.

Lee, S., Kim, S., 2011, Heuristic method of load leveling for optimal production in shipyard, *Proceeding of Autumn Korean Institute of Industrial Engineers*, pp. 183-189.

Roh, M.-I., Cha, J.-H., 2011, A block transportation scheduling system considering a minimisation of travel distance without loading of and interference between multiple transporters, *International Journal of Production Research*, 49(11), pp. 3231-3250.

손정열, 서흥원, 하병현, 2014, 조선 산업의 블록 적치장 운영계획 휴리스틱 알고리즘, *Journal of the Society of Naval Architects of Korea*, 51(3), pp. 239-245.

Park, J., Kim, M., 2020, Optimization of quantity allocation using integer linear programming in shipbuilding industry, *Journal of the Society of Naval Architects of Korea*, 57(1), pp. 45-51.

Pinedo, M. L. (2012). *Scheduling*, Springer.

Kim, D.-W., Na, D.-G., Chen, F. F., 2003, Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective, *Robotics and Computer-Integrated Manufacturing*, 19(1-2), pp. 173-181.

Chen, C.-L., Chen, C.-L., 2009, Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, 43(pp. 161-169.

Lee, J.-H., Yu, J.-M., Lee, D.-H., 2013, A tabu search algorithm for unrelated parallel machine scheduling with sequence-and machine-dependent setups: minimizing total tardiness, *The International Journal of Advanced Manufacturing Technology*, 69(pp. 2081-2089.

Chaudhry, I. A., Elbadawi, I. A., 2017, Minimisation of total tardiness for identical parallel machine scheduling using genetic algorithm, *Sādhanā*, 42(1), pp. 11-21.

Lee, C.-H., 2018, A dispatching rule and a random iterated greedy metaheuristic for identical parallel machine scheduling to minimize total tardiness, *International Journal of Production Research*, 56(6), pp. 2292-2308.

Park, J., Chun, J., Kim, S. H., Kim, Y., Park, J., 2021, Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning, *International Journal of Production Research*, 59(11), pp. 3360-3377.

Zhang, Z., Zheng, L., Weng, M. X., 2007, Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning, *The International Journal of Advanced Manufacturing Technology*, 34(pp. 968-980.

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., Hu, K., 2012, Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning, *Computers & operations research*, 39(7), pp. 1315-1324.

Yuan, B., Jiang, Z., Wang, L., 2016, Dynamic parallel machine scheduling with random breakdowns using the learning agent, *International Journal of Services Operations and Informatics*, 8(2), pp. 94-103.

Paeng, B., Park, I.-B., Park, J., 2021, Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups, *IEEE Access*, 9(pp. 101390-101401.

Julaiti, J., Oh, S. C., Das, D., Kumara, S., 2022, Stochastic parallel machine scheduling using reinforcement learning, *Journal of Advanced Manufacturing and Processing*, 4(4), pp. e10119.

Li, F., Lang, S., Hong, B., Reggelin, T., 2023, A two-stage RNN-based deep reinforcement learning approach for solving the parallel machine scheduling problem with due dates and family setups, *Journal of Intelligent Manufacturing*, pp. 1-34.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement learning: An introduction*, MIT press.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., 2015, Human-level control through deep reinforcement learning, *nature*, 518(7540), pp. 529-533.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*, pp.

Dagkakis, G., Heavey, C., 2016, A review of open source discrete event simulation software for operations research, *Journal of Simulation*, 10(3), pp. 193-206.

Thomas, T. E., Koo, J., Chaterji, S., Bagchi, S., 2018, Minerva: A

reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations, *2018 10th international conference on communication systems & networks (COMSNETS)*, IEEE.

Jang, I., Kim, D., Lee, D., Son, Y., 2018, An agent-based simulation modeling with deep reinforcement learning for smart traffic signal control, *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE.

Pinciroli, L., Baraldi, P., Compare, M., Esmaeilzadeh, S., Farhan, M., Gohre, B., Grugni, R., Manca, L., Zio, E., 2020, Agent-based modeling and reinforcement learning for optimizing energy systems operation and maintenance: the Pathmind solution, *Proceedings of the 30th European safety and reliability conference and the 15th probabilistic safety assessment and management conference*, Research Publishing, Singapore.

Greasley, A., 2020, Implementing reinforcement learning in simio discrete-event simulation software, *Proceedings of the 2020 Summer Simulation Conference*.

Rabe, M., Dross, F., Wuttke, A., Duarte, A., Juan, A., Lourenço, H., 2017, Combining a discrete-event simulation model of a logistics network with deep reinforcement learning, *Proceedings of the MIC and MAEB 2017 Conferences. July 4th-7th, Barcelona, Spain*.

Shiue, Y.-R., Lee, K.-C., Su, C.-T., 2018, Real-time scheduling for a smart factory using a reinforcement learning approach, *Computers & Industrial Engineering*, 125(pp. 604-614.

Mayer, S., Classen, T., Endisch, C., 2021, Modular production control using deep reinforcement learning: proximal policy optimization, *Journal of Intelligent Manufacturing*, 32(8), pp. 2335-2351.

Pires, F., Ahmad, B., Moreira, A. P., Leitão, P., 2021, Recommendation system using reinforcement learning for what-if simulation in digital twin, *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, IEEE.

Preston, R. A. (2017). *Utilizing a discrete event simulation of material handling plans to calculate reinforcement learning rewards*, The University of Alabama in Huntsville.

Stricker, N., Kuhnle, A., Sturm, R., Friess, S., 2018, Reinforcement learning for adaptive order dispatching in the semiconductor industry, *CIRP Annals*, 67(1), pp. 511-514.

Menda, K., Chen, Y.-C., Grana, J., Bono, J. W., Tracey, B. D., Kochenderfer, M. J., Wolpert, D., 2018, Deep reinforcement learning for event-driven multi-agent decision processes, *IEEE Transactions on Intelligent Transportation*

Systems, 20(4), pp. 1259-1268.

Woo, J. H., Cho, Y. I., Nam, S. H., Nam, J.-H., 2021, Development of a reinforcement learning-based adaptive scheduling algorithm for block assembly production line, *2021 Winter Simulation Conference (WSC)*, IEEE.

Nam, S.-H., Oh, S.-H., Yoon, H.-C., Cho, Y.-I., Cho, K.-Y., Kwak, D.-H., Woo, J. H., 2022, Development of Des Application for Factory Material Flow Simulation With Simpy, *2022 Winter Simulation Conference (WSC)*, IEEE.

Lee, Y. H., Pinedo, M., 1997, Scheduling jobs on parallel machines with sequence-dependent setup times, *European Journal of Operational Research*, 100(3), pp. 464-474.

Abstract

Nam So Hyun
Naval Architecture and Ocean Engineering
The Graduate School
Seoul National University

The profile shops in shipyards produce section steels required for block production of ships. With recent increases in shipyard orders, along with the need for productivity improvement, various measures such as optimizing production plans, increasing outsourcing volume, and introducing new equipment have been considered. However, solely relying on new equipment may lead to short-term solutions due to significant initial investments and potential cyclical downturns in the shipbuilding industry. Moreover, as the shipyard's in-house production capacity is limited, further increasing outsourcing is practically constrained. Hence, considering cost and sustainability, production planning optimization emerges as a viable alternative to enhance productivity.

Although numerous studies have used heuristics, meta-heuristics, and Integer Linear Programming in shipbuilding production to optimize production plans, research on the parallel machine scheduling problem, specifically in the profile shop, has been relatively scarce. While parallel machine scheduling research has been conducted in other manufacturing fields, the existing studies often neglect real-world variability and focus on single objective

functions, overlooking the decision-making process involving multiple objectives. To address these gaps, this study aims to perform scheduling optimization research on the parallel machine scheduling problem in shipyards, effectively incorporating real-world production environment variability and considering multiple objectives that are relevant to actual managerial decision-making.

Specifically, we propose a dynamic scheduling algorithm for enhancing productivity in the profile shop, taking into account the variability in job assignments and job durations, while minimizing tardiness and setup change frequency as objective functions. For the task sequencing problem in the profile shop, we define a Markov Decision Process model that considers both objective functions, and then employ the proximal policy optimization algorithm, a policy-based reinforcement learning technique, to learn the optimal scheduling policy. The performance of the developed algorithm is evaluated by comparing it with priority rules (SSPT, ATCS, MDD, COVERT rule) using sampled performance data in test scenarios. The results confirm that the proposed algorithm outperforms the priority rules when considering both average setup frequency and average tardiness as comprehensive metrics.

Finally, we investigate the applicability of the proposed scheduling algorithm to other parallel machine problems in shipyards by experimenting with a general parallel machine scheduling problem formulated using probability distributions. The results show that the proposed scheduling algorithm effectively achieves the objectives of minimizing setup time and tardiness compared to other priority rules

in this general problem.

Keywords : Reinforcement Learning, Parallel Machine Scheduling Problem, Discrete Event Simulation, Dynamic Scheduling
Student Number : 2021-25405