

## 석사학위논문

# 조선소 내 블록 적치장의 효율적인 운용을 위한 강화학습 환경 설계 및 평가

최국철

부산대학교 대학원  
정보융합공학과

2023년 2월

# 조선소 내 블록 적치장의 효율적인 운용을 위한 강화학습 환경 설계 및 평가

이 논문을 공학석사 학위논문으로 제출함

최국철

부산대학교 대학원

정보융합공학과

지도교수 김종덕

최국철의 공학석사 학위논문을 인준함

2022년 월 일

위원장: 김원석 인

위원: 정상화 인

위원: 김종덕 인

# 목 차

<b>제 1 장 서론</b>	<b>1</b>
1.1 연구배경 . . . . .	1
1.2 연구문제 진술 . . . . .	4
1.3 연구 기여 . . . . .	7
<b>제 2 장 관련 연구</b>	<b>10</b>
2.1 조선소 내 블록 배치 알고리즘 . . . . .	10
2.2 심층강화학습 (Deep Reinforcement Learning) . . . . .	12
2.2.1 가치 기반 방법 (Value-based method) . . . . .	15
2.2.2 정책 기반 방법 (Policy-based method) . . . . .	16
2.2.3 액터-크리틱 방법 (Actor-critic method) . . . . .	18
<b>제 3 장 블록 배치 환경 (Block Arrangement Environment)</b>	<b>21</b>
3.1 블록 운반 작업 시뮬레이터 . . . . .	24
3.1.1 블록 운반 작업 스케줄 . . . . .	24
3.1.2 블록 적치 작업 . . . . .	26
3.1.3 블록 반출 작업 . . . . .	27
3.1.4 블록 재배치 작업 . . . . .	28
3.2 강화학습 구성 요소 . . . . .	29
3.2.1 상태 (State) . . . . .	29
3.2.2 행동 (Action) . . . . .	29
3.2.3 보상 (Reward) . . . . .	30
<b>제 4 장 실험 및 성능 평가</b>	<b>32</b>
4.1 강화학습 알고리즘별 학습 성능 분석 . . . . .	36
4.2 블록 적치장 포화율에 따른 학습 성능 분석 . . . . .	39
4.3 블록 적치 기간 정보에 따른 학습 성능 분석 . . . . .	43
4.4 보상 신호 조합에 따른 학습 성능 분석 . . . . .	46
<b>제 5 장 결론</b>	<b>49</b>

# 그림 목록

1.1	블록 단위로 제작되는 선박 건조 작업 . . . . .	1
1.2	반출 작업 경로 확보를 위한 재배치 작업 예시 . . . . .	4
2.1	MDP기반 환경과 에이전트를 이용한 강화학습 프로세스 . . . . .	12
2.2	심층강화학습 알고리즘의 학습 원리 . . . . .	14
3.1	블록 배치 환경기반 강화학습 프로세스 . . . . .	22
3.2	블록 운반 작업 스케줄 예시 . . . . .	26
4.1	에이전트에 따라 10,000번의 시뮬레이션동안 스케줄 내 블록 운반 작업을 모두 완료한 에피소드 수. 오차막대는 95% 신뢰구간을 나타낸다. . . . .	37
4.2	에이전트에 따라 10,000번의 시뮬레이션동안 발생한 재배치 작업 횟수 . . . . .	37
4.3	PPO 알고리즘기반 블록 배치 예제 . . . . .	38
4.4	블록 적치장 포화율에 따른 학습 곡선 . . . . .	39
4.5	적치장 포화율에 따라 10,000번의 시뮬레이션동안 스케줄 내 블록 운반 작업을 모두 완료한 에피소드 수 . . . . .	40
4.6	블록 적치장 포화율 별 재배치 작업 발생 횟수 . . . . .	41
4.7	블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드 수 . . . . .	44
4.8	블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 발생한 재배치 작업 횟수 . . . . .	45
4.9	블록 적치장 포화율에 따른 학습 곡선 . . . . .	46
4.10	보상 신호 조합에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드 수 . . . . .	47
4.11	보상 신호 조합에 따라 10,000만번의 시뮬레이션동안 발생한 재배치 작업 횟수 . . . . .	48

# 조선소 내 블록 적치장의 효율적인 운용을 위한 강화학습 환경 설계 및 평가

최 국 철

부산대학교 AI대학원 정보융합공학과

## 요약

조선소에서 건조되는 선박은 다수의 블록으로 나누어 만들어진다. 각 블록은 다양한 공정을 거쳐 완성되며, 수많은 블록의 공정 작업이 동시에 진행되기 때문에 모든 블록의 공정 작업 스케줄을 정확히 맞추는 것은 어려운 일이다. 따라서 블록이 다음 공정 작업을 바로 진행하지 못하고 대기해야 하는 상황이 빈번히 발생한다. 블록의 임시 대기 장소를 블록 적치장이라고 하며, 이곳에서 다양한 블록 운반 작업이 발생한다. 블록 운반 작업은 많은 시간 및 금전적 비용이 소모되므로 적치장에서 발생하는 불필요한 블록 재배치 작업은 큰 손실을 야기한다. 따라서 블록 재배치 작업을 최소화할 수 있도록 블록을 배치하는 것은 효율적인 블록 적치장 운용을 위해 가장 중요한 요소이다. 본 연구는 재배치 작업 최소화를 위한 블록 배치 전략을 학습하기 적합한 강화학습 환경인 블록 배치 환경을 제안한다. 블록 배치 환경은 적치장에서 발생하는 블록 운반 작업 시뮬레이션을 제공하고, 강화학습 에이전트는 블록 배치 환경과 상호작용하며 시행착오를 통해 블록 배치 전략을 학습한다. 각 조선소에 적합한 블록 배치 전략 학습을 위해 고려해야 할 요소를 정의하고, 다양한 시나리오에 대한 실험 및 분석을 통해 블록 배치 환경이 블록 배치 전략을 학습하기 적합한 환경이라는 것을 보인다.

# 제 1 장 서론

## 1.1 연구배경

조선이란 설계, 제조, 가공 및 조립의 과정을 거쳐 선박을 건조하는 것을 말한다. 조선 산업은 다량의 대형 설비 및 숙련된 노동력이 요구되는 대표적인 노동 집약 산업이다. 최근 4차 산업혁명으로 인해 사물인터넷 (Internet of Things), AI, 5G 등의 기술이 발전하며, 많은 조선소가 첨단 IT 기술을 조선업에 접목하여 기술 경쟁력을 높이기 위해 노력하고 있다. 특히, 선박 건조 과정에서 비용 절감 및 생산성 향상을 위해 스마트 야드를 구축하려는 노력이 활발히 이루어지고 있다. 야드는 조선소 내 선박 건조에 사용되는 모든 공간을 나타낸다. 스마트 야드는 제한적인 야드 공간에서 선박 건조 과정을 최적화하기 위해 IT 기술을 활용하여 선박 건조 과정 및

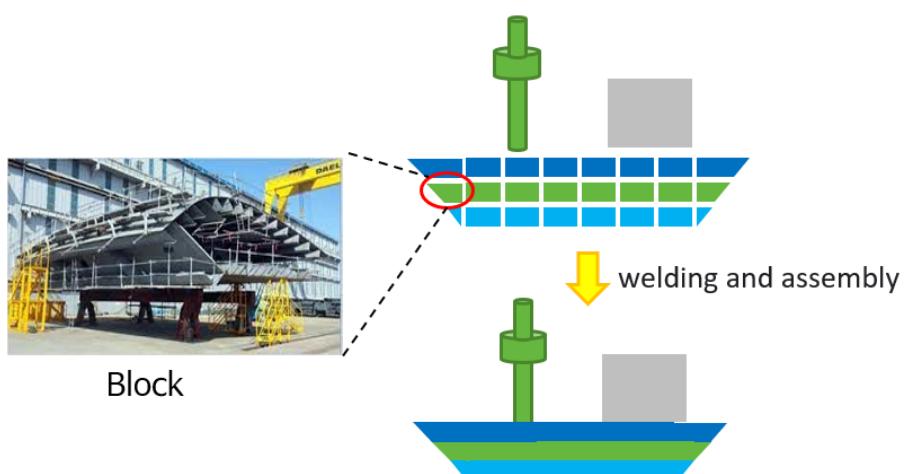


그림 1.1: 블록 단위로 제작되는 선박 건조 작업

야드를 관리하는 것을 의미한다.

조선소에서 건조되는 선박은 매우 크기 때문에 그림 1.1과 같이 선체를 다수의 블록으로 나누어 만들고 마지막 단계에 조립하여 완성한다. 블록은 배에 따라 다양한 모양과 크기를 가지며, 주로 철판 가공에서 시작하여 의장, 도장, P.E. 등의 공정 작업을 거쳐 최종 선박 조립에 이용된다. 블록의 공정 소요 시간은 작업자의 숙련도 및 블록의 특성에 따라 다르며, 야드 내에서 수많은 블록의 공정 작업이 동시에 진행되기 때문에 모든 블록의 공정 작업 스케줄을 정확히 맞추는 것은 굉장히 어렵다. 이러한 이유로 공정 작업이 완료된 블록이 다음 공정을 바로 진행하지 못하고 대기해야 하는 상황이 빈번히 발생한다.

블록이 다음 공정을 바로 진행하지 못하고 임시로 운반 및 적치되어 다음 공정을 대기하는 장소를 블록 적치장이라고 한다. 블록 적치장에서 발생하는 주요 블록 운반 작업으로 블록 적치 작업과 블록 반출 작업이 있다. 블록 적치 작업은 블록이 다음 공정을 진행하지 못하고 대기해야 하는 상황에 발생하며, 블록 반출 작업은 블록 적치장에서 대기 중인 블록이 다음 공정에 투입될 때 발생한다. 블록 반출 작업을 통해 공정 일정에 맞추어 적치장에서 대기 중인 블록을 공정 장소로 운반하는 작업은 매우 중요하다. 반출 작업이 늦어질 경우, 해당 블록의 남은 공정 작업이 함께 늦춰지고 타 블록의 공정 작업에도 영향을 끼쳐, 선박 건조 일정 전체가 연기되는

문제가 발생할 수 있기 때문이다.

블록 반출 작업이 발생할 경우, 반출 경로상 다른 블록이 존재하여 해당 블록을 다른 위치로 옮긴 후 반출 작업을 해야 하는 경우가 빈번히 발생한다. 반출 작업 시 방해되는 블록을 간섭 블록이라고 부른다. 적치장에 많은 블록이 적치되어 있을수록 반출 작업 시 간섭 블록이 존재할 가능성이 높으며, 간섭 블록이 많을수록 불필요한 블록 재배치 작업 증가로 인해 반출 작업이 늦어지게 된다.

## 1.2 연구문제 진술

블록 재배치 작업은 적치장 내 블록 운반 작업을 증가시키는 주요 원인이다. 블록은 그림 1.1에서 보이는 것과 같이 매우 크고 무겁다. 따라서 블록 운반 작업은 트랜스포터라고 불리는 운송 수단을 이용하고, 4명 이상의 신호수 및 작업자가 동반되어 많은 시간 및 금전적 비용이 소모되는 작업이다. 그러므로 블록 재배치 작업을 줄일 수 있도록 적절히 블록을 배치하는 것은 효율적인 블록 적치장 운용을 위해 반드시 필요하다.

그림 1.2는 6x5 사각형 그리드 공간의 블록 적치장에서 반출 작업 시 반출 경로를 확보하기 위해 발생하는 블록 재배치 작업의 예시를 나타낸다. 블록의 크기는 1x1로 가정하며, 블록의 숫자는 남은 적치 기간을 의미한다. 마지막 행을 제외한 5x5 공간이 블록 적치에 이용되며, 블록 적치장 왼쪽 아래에 트랜스포터 출입구가 있는 상황을 가정하여 해당 위치를 출입구로 표기하였다. 모든 블록은 트랜스포터에 탑재되어 출입구를 통해 블록 적치장에 반입 및 반출된다. 그림 1.2는 반출 작업 대상 블록의 반출 가능

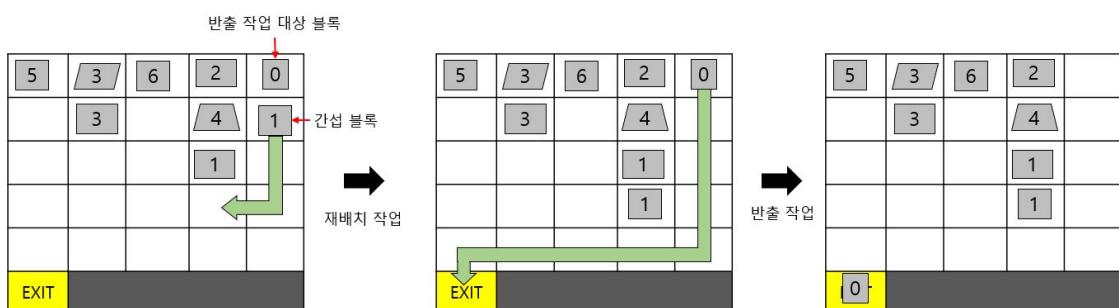


그림 1.2: 반출 작업 경로 확보를 위한 재배치 작업 예시

경로가 존재하지 않아 반출 경로 확보를 위해 간접 블록을 적치장 내 임의의 위치로 재배치한 후 반출 작업을 수행한다.

블록 적치장 내 블록 재배치 작업을 줄이기 위한 블록 배치 기술은 다양한 방법으로 연구되어 왔다. 블록 배치 문제는 [1]에서 최초로 소개되었다. 이후 해당 문제를 풀기 위하여 다양한 접근들이 제안되었으며 [1] - [5]는 블록 운반 작업에 대한 도메인 지식을 바탕으로 규칙 기반 블록 배치 휴리스틱 알고리즘을 제안했다. [2]는 블록 배치 문제를 수학적 프로그래밍 모델로 정의하고 [1]의 휴리스틱 알고리즘을 개선했다. [3]는 유전 알고리즘과 다이나믹 휴리스틱 알고리즘에 기반한 블록 배치 알고리즘을 제안했다. [4]는 bottom-left-fill (BLF) 방법 기반 블록 배치 알고리즘을 제안하였고, [5]는 타부 서치 알고리즘 기반의 블록 배치 알고리즘을 제안했다. [6]는 다양한 블록의 크기를 고려하여 그리디 알고리즘 기반의 블록 배치 알고리즘을 제안했다.

규칙 기반 휴리스틱 알고리즘은 현실의 문제에 존재하는 여러 시나리오를 고려하여 문제를 정형화하고, 각 시나리오에 적합한 규칙을 정의한 후, 정의된 규칙을 바탕으로 알고리즘을 설계한다. 하지만 블록 배치 문제와 같은 현실의 복잡한 문제에서 발생할 수 있는 다양한 시나리오를 모두 고려하여 문제를 정형화하는 것은 매우 어렵다. 게다가, 문제 내 시나리오의 수가 매우 많을 경우, 모든 시나리오를 정확히 처리하기 위한 규칙을 정

의하는 것 또한 매우 어렵다. 이러한 이유로, 알고리즘이 매우 복잡해지는 시나리오 및 요인은 문제를 정형화하는 과정에서 제외되는 경우가 존재한다. 즉, 규칙 기반 휴리스틱 알고리즘은 블록 배치 문제에 영향을 끼치는 다양한 요소를 반영하는 것이 어렵고, 제약 조건이 많이 존재할수록 실용성이 낮아지는 단점이 있다.

### 1.3 연구 기여

우리의 연구는 [7]에서 제안한 머신러닝(machine learning) 기반의 접근에서 동기를 얻었다. 머신러닝이란 명시적인 프로그래밍 없이 데이터를 바탕으로 컴퓨터를 학습시키는 AI 기법이다. 머신러닝 기반 접근은 문제에 관련된 다양한 데이터를 이용하여 복잡한 문제를 비교적 효율적이고 정확하게 해결할 수 있다. 머신러닝은 학습 메커니즘에 따라 크게 지도학습(supervised learning), 비지도학습(unsupervised learning), 강화학습(reinforcement learning)으로 분류된다. 블록 배치 문제는 불필요한 블록 운반 횟수 최소화를 위한 블록 배치 위치 결정 문제로 치환할 수 있다. 머신러닝 기법을 이용하여 블록 배치 의사 결정 문제에 접근하는 방법은 지도학습 또는 강화학습 기반 접근이 가능하다.

지도학습 기반 접근은 조선소 내 기존 작업자들의 노하우가 축적된 블록 운반 작업 데이터를 학습하는 것이다. 블록 배치 위치 결정 문제는 지도학습이 가지고 있는 한계점에 취약하다. 지도학습은 데이터와 정답 쌍을 함께 학습하여 학습되지 않은 데이터에 대한 정답을 예측한다. 지도학습의 예측 성능은 데이터의 양과 질에 매우 의존적이다. 따라서 데이터의 양이 충분하지 않거나 데이터에 대한 정답이 올바르지 않을 경우, 지도학습을 통해 최적의 의사 결정을 달성하는 것이 어렵다. 블록 배치 문제는 기존 작업자들에 의해 결정되는 블록 배치 데이터가 최적의 위치라는 보장이 없다.

따라서 이를 이용하여 학습할 경우 재배치 작업 최소화를 달성하는 것은 불가능하며, 기존 작업자들 이상의 블록 배치 성능을 기대하기 힘들다.

강화학습 기반 접근은 블록 운반 작업 시뮬레이터에서 생성되는 데이 터를 통해 블록 배치 행동 전략을 학습하는 방법이다. 현실의 문제를 강화 학습 방법으로 접근할 때, 가장 중요한 2가지 요소는 문제 해결에 적합한 강화학습 환경과 최종 목표를 달성할 수 있는 학습성능을 가진 강화학습 알고리즘이다. 강화학습 알고리즘은 거의 매년 최고 성능의 알고리즘이 개신될 정도로 활발히 연구되고 있으며, 다양한 분야에서 뛰어난 학습 성 능을 보이고 있다.

강화학습 알고리즘의 학습 성능이 아무리 뛰어나더라도 강화학습 환경 이 해결하고자 하는 문제에 적합하지 않을 경우 결코 최종 목표를 달성할 수 없다. 따라서 현실의 문제를 강화학습으로 접근할 때, 풀고자 하는 문제 상황을 잘 반영하여 시뮬레이션할 수 있는 강화학습 환경이 필수적이다. 하지만 문제에 영향을 끼치는 현실의 다양한 요인들을 모두 반영하여 시 물레이션 환경을 구축하는 것은 매우 어렵다. 그래서 우리는 블록 배치 문 제에서 고려되어야 하는 다양한 요인들을 반영한 강화학습 환경을 만드는 것에 집중하였다.

본 논문에서 우리는 블록 운반 작업 시뮬레이터 기반 블록 배치 강화학 습 환경을 제안한다. 블록 배치 강화학습 환경에서 에이전트는 블록 재배

치 작업을 최소화할 수 있는 블록 배치 전략을 학습할 수 있다. 우리 연구의 기여는 다음과 같다.

- 블록 적치장에서 발생하는 블록 운반 작업 시뮬레이션을 바탕으로 강화학습 통해 블록 배치 전략을 학습할 수 있는 환경을 제공한다.
- 블록 배치 전략 학습에 반영되어야 하는 요인들을 정의하고, 실험을 통해 해당 요인들이 학습에 끼치는 영향을 분석한다.
- 각 조선소의 특성에 맞게 블록 배치 전략을 학습할 수 있도록 블록 운반 데이터를 이용하여 맞춤 데이터를 생성하는 블록 운반 작업 스케줄 생성기를 제공한다.
- 추후 연구자들이 블록 배치 강화학습 환경을 이용하여 새로운 블록 배치 알고리즘의 성능을 평가할 수 있도록 기준 성능(baseline performance)을 제공한다.

본 논문의 이후 구성은 다음과 같다. 2장에서는 블록 배치 문제 관련 기존 연구 및 심층강화학습을 설명한다. 3장에서는 블록 배치 강화학습 환경의 블록 운반 작업 시뮬레이션 프로세스와 강화학습 요소들에 대하여 설명한다. 4장에서는 블록 배치 문제에 영향을 끼치는 다양한 요인 정의 및 각 요인의 다양한 시나리오에 대한 학습 성능을 제공하고 분석한다. 마지막 장은 연구의 결론과 추후 연구에 대하여 설명한다.

## 제 2 장 관련 연구

### 2.1 조선소 내 블록 배치 알고리즘

블록 적치장 내 블록 재배치 작업을 최소화하기 위한 블록 배치를 위해 다양한 알고리즘이 제안되었다. [1]에서 최초로 블록 적치장 내 블록 배치 문제를 정의하고 규칙 기반 휴리스틱 알고리즘을 제안하였다. [2]는 해당 문제를 ABSLAP(assembly block storage location assignment problem)로 정의하고, [1]의 휴리스틱 알고리즘을 개선하였다. [3]은 PSLAP(planar storage location assignment problem)로 해당 문제를 정의하였으며 유전 알고리즘과 다이내믹 PSLAP 휴리스틱 알고리즘을 제안하고 성능 비교를 통해 다이내믹 PSLAP 휴리스틱 알고리즘의 블록 배치 성능을 증명했다. [4], [5]는 블록 배치 순서와 위치를 모두 고려한 블록 배치 휴리스틱 알고리즘을 제안하였으며, BLF 알고리즘[4]과 타부 서치 알고리즘[5] 기반의 블록 배치 휴리스틱 알고리즘이 제안했다. BLF 알고리즘은 주어진 공간에서 가장 아래쪽의 왼쪽부터 빈공간에 객체를 채우는 알고리즈다. [7]에서는 블록 배치 문제를 마르코프 결정 과정(Markov decision process, MDP)으로 정의하여 머신러닝 기법 중 하나인 강화학습기반 블록 배치 전략을 제안했다. 심층강화학습 알고리즘인 비동기적 어드밴티지 액터-크리

틱(Asynchronous advantage actor-critic) 알고리즘을 이용하여 재배치 작업 최소화를 위한 블록 배치 전략을 학습하였으며, BLF, PSLAP 휴리스틱 알고리즘과 블록 배치 성능을 비교하여 심층강화학습기반 블록 배치 성능의 우수성을 증명하였다.

## 2.2 심층강화학습 (Deep Reinforcement Learning)

강화학습은 학습의 주체인 에이전트가 주어진 환경과 상호작용하며 시행착오를 통해 문제에 적합한 액션 전략을 학습하는 머신러닝 기법이다[8]. 강화학습의 에이전트는 문제에 대한 어떠한 사전 지식도 없는 상태에서 환경을 관찰하고, 액션을 수행한 후 얻는 보상 신호를 통해 학습한다. 에이전트의 최종 목적은 보상의 총합을 최대화하는 전략을 수립하는 것이다. 최근, 강화학습은 다양한 분야의 어려운 문제들에 적용되어 우수한 결과를 보이고 있다. 이와 같은 성과를 보이게 된 결정적인 계기는 강화학습에 딥러닝 기술을 적용한 심층강화학습의 등장 덕분이다[9].

딥러닝(Deep Learning)은 인공신경망 기반의 머신러닝 기법 중 하나이며, [10]에서 역전파 알고리즘을 이용한 신경망 학습 방법을 제안하며 처음 등장하였다. 딥러닝을 적용한 최초의 심층강화학습 알고리즘인 Deep Q Network(DQN)은 Atari 게임 화면의 원본 이미지를 학습에 이용하여 인간

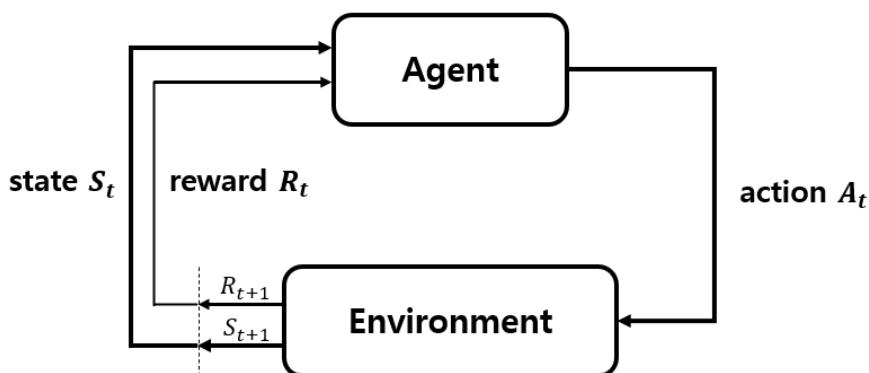


그림 2.1: MDP기반 환경과 에이전트를 이용한 강화학습 프로세스

수준의 게임 플레이 성능을 보였다[9]. 강화학습에 딥러닝 기술이 접목되면서 등장한 신경망기반 에이전트는 복잡한 다차원의 입력 공간을 저차원의 특징 공간으로 변환하는 지각능력을 가질 수 있게 되었다. 즉 심층강화학습의 에이전트는 다차원 입력 공간을 가지는 복잡한 문제를 전처리 없이 학습할 수 있다.

전통적으로, 강화학습은 순차적 의사 결정 문제에 적용하기 적합하며, 순차적 의사 결정 문제는 마르코프 결정 과정(Markov decision process, MDP)으로 정의한다. MDP는 튜플  $(S, A, P, R, \gamma)$ 로 순차적 의사 결정 문제를 정의할 수 있다. MDP안에서  $S$ 는 모든 상태  $s$ 의 집합을 의미하며,  $A$ 는 가능한 모든 액션  $a$ 의 집합을 의미한다.  $P(s'|s, a)$ 는 상태  $s$ 에서 액션  $a$ 를 수행할 때, 상태  $s'$ 으로 전이될 확률을 의미하며,  $R(s'|s, a)$ 은 상태  $s$ 에서 액션  $a$ 를 수행하여  $s'$ 으로 전이될 때 얻게되는 보상의 기댓값을 나타낸다.  $\gamma$ 는 현재 얻게되는 보상과 미래에 얻게될 보상의 중요도를 정하는 할인 인자(discount factor)이다.

그림 2.1은 MDP안에서 에이전트가 환경과 상호작용하며 학습하는 구조이다. 에이전트는 상태  $S_t$ 에서 액션  $A_t$ 를 수행한다. 그러면 환경은 다음 상태에 해당하는 상태  $S_{t+1}$ 과 보상 신호  $R_{t+1}$ 을 에이전트에게 반환한다. 에이전트는 이 과정을 반복하며 각 상태에 대해 미래에 얻게될 보상의 총 합을 최대화하는 액션 전략을 학습하게 된다.

에이전트가 학습을 통해 수립하는 액션 전략을 정책(policy)이라고 하며  $\pi$ 로 표기한다.  $\pi(s)$ 는 에이전트가 정책  $\pi$ 를 따를 때, 상태  $s$ 에서 선택되는 액션  $a$ 를 나타내며,  $\pi(a|s)$ 는 정책  $\pi$ 를 따를 때, 상태  $s$ 에서 액션  $a$ 가 선택될 확률을 나타낸다. 에이전트의 목표는 다른 모든 정책과 비교하여 미래에 얻게 될 누적 보상의 합이 항상 크거나 같은 것을 보장하는  $\pi^*$ 을 찾는 것이다.  $\pi^*$ 는 아래 수식으로 나타낸다.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{H-1} \gamma^k r_{k+1} | s_0 = s \right] \quad (2.1)$$

위 수식에서  $k$ 는 MDP안에서 현재 스텝과 미래 스텝간 거리를 나타내고,  $H$ 는 MDP안에서 발생하는 전체 스텝의 수를 의미한다.  $\gamma$ 는 미래에 얻게 될 보상의 가치를 조절하는 감쇠 인자이고,  $r_k$ 는  $k$  스텝 이후에 얻게 될 보상을 나타낸다.

심층강화학습은 딥러닝과 강화학습이 융합된 머신러닝 기술이다. 딥러

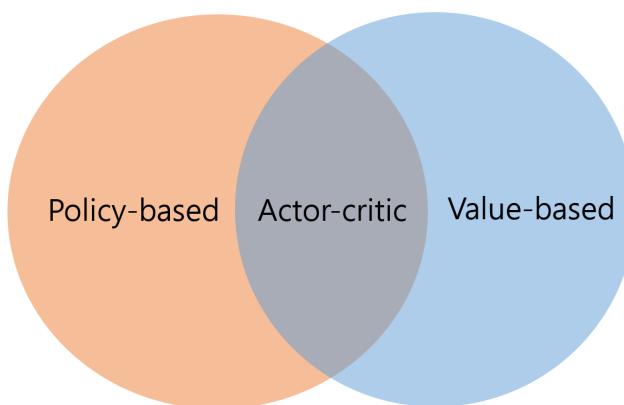


그림 2.2: 심층강화학습 알고리즘의 학습 원리

닝 학습 메커니즘은 강화학습 에이전트에게 복잡한 문제를 해결할 수 있는 능력을 가지게 한다. 심층강화학습의 알고리즘은 학습 원리에 따라 크게 세 종류로 분류된다. 그림 2.2은 심층강화학습의 세가지 학습 원리를 나타낸 것으로 가치 함수(value function)을 이용하는 가치 기반 방법, 직접 정책을 학습하는 정책 기반 방법, 가치 함수와 정책을 모두 학습하는 방식인 액터-크리틱 방법이 있다.

### 2.2.1 가치 기반 방법 (Value-based method)

가치 기반 방법은 상태 또는 상태-행동 쌍(state-action pair)에 대한 가치를 기준으로 에이전트의 행동을 결정하는 방식이다. 가치란 특정 상태 또는 상태-행동 쌍으로부터 미래에 얻게될 누적 보상의 합을 의미한다. 가치가 정확히 학습될 경우, 매 스텝 가장 높은 가치의 액션을 결정하는 것만으로 최적 정책이 된다. 정책  $\pi$ 를 따를 때, 상태  $s$ 의 가치는  $V_\pi(s)$ 로 표기하며, 아래의 수식으로 정의할 수 있다.

$$V_\pi(s) = \mathbb{E}_\pi \left[ \sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s \right] \quad (2.2)$$

정책  $\pi$ 를 따르고, 상태  $s$ , 행동  $a$ 를 선택할 때 상태-행동 쌍의 가치는  $Q_\pi(s, a)$ 로 표기하며, 아래의 수식으로 정의할 수 있다.

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{H-1} \gamma^k r_{k+1} | s_0 = s, a_0 = a \right] \quad (2.3)$$

심층강화학습의 가치함수  $V_\pi(s)$ 와  $Q_\pi(s, a)$ 는 일반적으로 인공 신경망(neural network)으로 구성되어 있다. 최적의 가치함수를 만들기 위해 딥러닝 기법에 기반하여 인공 신경망의 파라미터를 지속적으로 업데이트한다. 가치 기반 방법의 대표적인 심층강화학습 알고리즘으로 DQN이 있다.

DQN은 합성곱 신경망(Convolutional network)를 이용하여 게임의 원본 이미지만으로 가치함수  $Q_\pi(s, a)$ 를 최적화한다. [9] 이전에는 샘플간 높은 상관관계와 같은 목적 가치 변화 문제로 딥러닝 기법을 강화학습에 적용하지 못했다. DQN은 experience replay 기법으로 샘플간 상관관계를 줄이고, 실제 액션을 수행하는 Q-network와 target network를 별도로 구축하여 target value가 자주 바뀌지 않도록 하였다. 결국 DQN은 experience replay 기법과 target network를 이용하여 CNN을 통해  $Q_\pi(s, a)$ 를 최적화하는데 성공한 최초의 심층강화학습 알고리즘이다.

### 2.2.2 정책 기반 방법 (Policy-based method)

가치 함수를 이용하지 않고 직접 에이전트의 행동 정책을 학습을 통해 최적화하는 방법을 정책 기반 또는 정책 경사 (policy-gradient) 방법이라고 한다. 정책이란 에이전트가 관찰한 환경을 바탕으로 행동을 선택하는

전략이다. 가치 기반 방법은 따로 정책을 학습하지 않지만 가치 함수를 이용하여 매 순간 가장 높은 가치의 행동을 선택하는 암묵적인 정책이 존재 한다. 정책 기반 방법은 무수히 많은 행동이 존재하는 연속적인 행동 공간 (continuous action space)에서 가치 기반 방법보다 더 나은 학습 성능을 보인다. 연속적인 행동 공간에서는 가치 기반 방법의 학습 성능이 더 낮은 이유는 학습해야 할 상태-행동 쌍이 무한에 가까워 최적화하기 어렵기 때문이다.

심층강화학습에서 정책은 인공 신경망에 기반하여 매개변수화 되어 있으며  $\pi_\theta$ 로 표현한다.  $\theta$ 는 인공 신경망을 구성하는 매개변수들을 의미한다.  $\pi_\theta$ 를 최적화하기 위해 정책 경사 방법이 이용되며, 정책 경사 방법은 누적 보상의 합이 최대화 되도록 경사 상승(gradient ascent) 방법으로 매개변수를 갱신한다.

REINFORCE 알고리즘은 가장 대표적인 정책 기반 방법의 강화학습 알고리즘이다[11]. 한 에피소드에서  $t$  시점의 누적 보상 예측값은  $G_t = \sum_{k=0}^{H-1} \gamma^k r_{k+t+1}$ 이며, REINFORCE 알고리즘에서 매개변수  $\theta$ 의 갱신은 아래 수식으로 정의한다.

$$\theta_{t+1} = \theta_t + \alpha \gamma^t G_t \nabla \log \pi_\theta(a_t | s_t) \quad (2.4)$$

$\alpha$ 는  $\theta$ 의 갱신 속도를 조절하기 위한 변수이다. REINFORCE 알고리즘

은 매 에피소드 완료 시 에피소드 내 모든 단계(step)에서 각 매개변수의 경사를 계산하고 식4를 바탕으로  $\theta$ 를 갱신한다.

### 2.2.3 액터-크리틱 방법 (Actor-critic method)

액터-크리틱 방법은 가치 기반 방법과 정책 기반 방법이 결합된 학습 방법이다. 액터-크리틱에서 액터는 실제 행동을 결정하는 에이전트의 정책을 의미하고, 크리틱은 정책을 평가하는 가치 함수를 뜻한다. 크리틱은 액터의 행동 이후 얻어진 상태를 바탕으로 에이전트의 행동을 평가한다. 액터, 크리틱 모두 독립적인 인공 신경망으로 구성되어 따로 학습된다. 액터는 정책 기반 방법의 target network인  $\pi^*$ 를 최적화하고, 크리틱은 가치 기반 방법의 target network인  $Q^*$ 를 최적화한다. 최근, 액터-크리틱 방법은 많은 분야에서 가치 기반 방법과 정책 기반 방법보다 더욱 좋은 강화학습 성능을 보인다. [7]에서 블록 배치 전략을 학습하기 위해 대표적인 액터-크리틱 방법인 A3C(Asynchronous Advantage Actor Critic)를 이용했다. A3C 는 다수의 액터가 비동기적으로 시뮬레이션 및 학습을 수행하여 목표 네트워크를 최적화한다[12]. 본 연구에서는 액터-크리틱 방법으로 A3C 보다 학습 성능이 좋다고 평가된 PPO(Proximal policy optimisation) 알고리즘을 실험에 이용한다.

PPO 간단한 구현만으로 뛰어난 학습 성능을 증명한 액터-크리틱 방법

이다[13]. 정책 기반 방법은 경사 예측(gradient estimating)을 통해 정책을 최적화한다. 이 때, 모델의 학습 성능은  $\theta$  갱신 크기에 굉장히 의존적이다. 갱신 크기를 작게 하면 학습 속도가 느린 문제가 있고, 갱신 크기를 크게하면 오실레이션이 발생하는 문제가 있다[14]. PPO는 이 문제를 해결하기 위해 목적 함수에 clipping 메커니즘을 적용했다. PPO는 확률적 경사 상승(stochastic gradient ascent)을 이용하여 Clipped 목적 함수(Clipped Objective function)를 최적화한다.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right] \quad (2.5)$$

위 식의  $r_t(\theta)$ 는 probability ratio를 뜻하며 아래 수식으로 정의한다.

$$r_t(\theta) = \frac{\pi_{\theta}(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)} \quad (2.6)$$

위 식의  $\pi_{\theta}$ 는 매개변수  $\theta$ 로 구성된 정책이다.  $\pi_{\theta}$ 를 최적화하기 위해서  $\pi_{\theta_{old}}$ 으로부터 샘플 데이터를 수집하고, 수집된 데이터를 이용하여 여러 차례  $\pi_{\theta}$ 를 최적화하는 주요 샘플링(importance sampling) 기법을 사용한다. 여기서  $\pi_{\theta_{old}}$ 는 새로운 정책으로 주기적으로 갱신된다.  $\hat{\mathbb{E}}$ 는 샘플에 대해서 경험적으로 얻어진 추정치의 기댓값을 의미하고,  $\hat{A}$ 는 이익 함수(advantage

function)이며 아래 수식으로 정의할 수 있다.

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (2.7)$$

식 3에서 이익 함수는  $Q^\pi(s, a)$ 와  $V^\pi(s)$ 의 차이를 이용하여 상태에 대한 행동의 가치를 측정한다. 식 1의 Clipped 목적 함수는 probability ratio의 이동 범위를  $[1 - \epsilon, 1 + \epsilon]$ 로 한정시킴으로써 지나치게 큰 정책 갱신을 방지 한다. 이러한 방식으로, 행동들에 대한 긍정적인 행동과 부정적인 행동에 대해서 너무 빠르게 학습되는 것을 피할 수 있다.

### 제 3 장 블록 배치 환경 (Block Arrangement Environment)

블록 배치 환경은 블록 운반 작업 시뮬레이터 기반 강화학습 환경이다. 블록 배치 환경의 목표는 에이전트가 블록 재배치 작업을 최소화할 수 있는 블록 배치 전략을 학습하는 것이다. 강화학습은 시행착오를 통해 최적의 행동 전략을 탐색한다. 블록 운반 작업은 많은 비용과 시간이 소모되는 작업이므로 실제 환경에서 시행착오를 통해 블록 배치 전략을 학습하는 것은 큰 비용을 소모해야 한다. 우리의 블록 배치 환경은 이산 사건 시뮬레이션기반 블록 운반 작업 시뮬레이터를 바탕으로 짧은 시간에 수많은 시행착오를 겪으며 학습할 수 있다. 강화학습 환경은 학습을 위해 매 스텝 에이전트로부터 행동을 받고, 현재 상태에 행동 반영 후 발생되는 새로운 상태와 보상을 에이전트에게 전달해야 한다. 따라서 강화학습 환경은 강화학습 주요 요소들인 상태, 행동, 보상을 정의해야 한다.

4.3는 블록 배치 환경과 에이전트가 상호작용하며 학습하는 프로세스를 나타낸 것이다. 블록 적치장 내 블록 배치 전략은 블록 적치장 및 블록 운반 작업 특성에 따라 달라질 수 있다. 따라서 강화학습을 현실에 적용하기 위해서는 실제 문제와 최대한 유사한 시뮬레이션 환경에서 학습하는 것이 중요하다. 우리의 블록 배치 환경은 이용자가 쉽게 커스터마이징할 수 있도록 블록 적치장, 블록 운반 작업 스케줄, 보상 신호등을 입력으로 받아

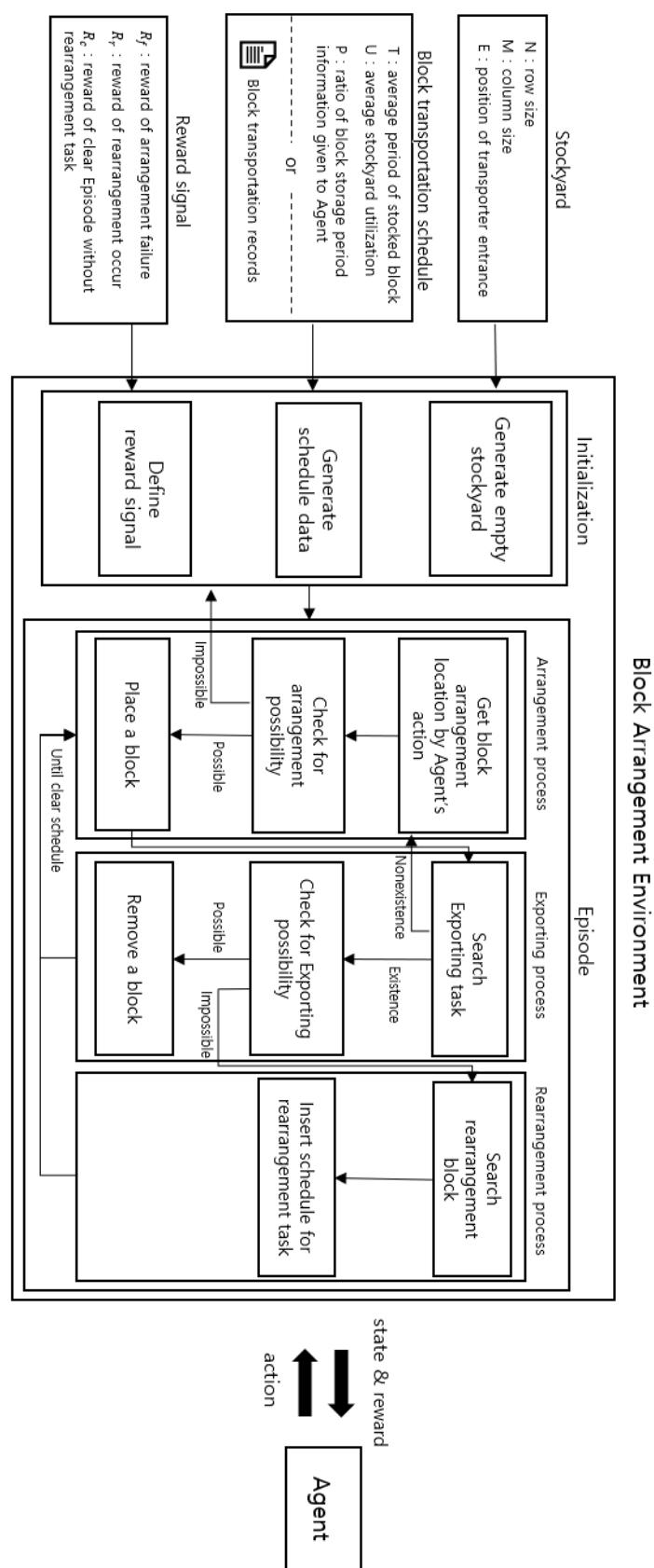


그림 3.1: 블록 배치 환경기반 강화학습 프로세스

서 이를 바탕으로 블록 적치장 및 블록 운반 작업 스케줄을 만들어 학습에 이용한다.

에피소드 시작 시 블록 운반 작업 스케줄과 빈 공간의 블록 적치장을 생성한 후, 스케줄 내 블록 운반 작업을 수행한다. 블록 운반 작업은 블록 적치 작업, 블록 반출 작업, 블록 재배치 작업으로 구분된다. 에이전트는 시뮬레이션안에서 블록의 적치 위치와 재배치 위치를 결정하고, 행동에 대한 보상을 받으며 최적의 블록 배치 전략을 학습하게 된다.

### 3.1 블록 운반 작업 시뮬레이터

블록 운반 작업 시뮬레이터는 블록 적치장내에서 발생하는 각종 블록 운반 작업 시뮬레이션을 제공한다. 블록 적치장은 가로 길이  $N$ , 세로 길이  $M$ 인 직사각형 모양이며, 적치장 외곽에 1개의 트랜스포터의 출입구가 존재하는 상황을 가정한다. 트랜스포터는 블록 적치장 내에서 상하좌우로 이동가능하며, 블록이 없는 빈 공간으로만 이동가능하다. 블록 운반 작업은 출발 지점에서 도착 지점까지 트랜스포터의 이동 가능 경로가 하나이상 존재하면 수행한다. 만일, 에이전트의 행동에 의해 결정된 블록 적치 위치 및 재배치 위치가 트랜스포터를 통해 이동이 불가능한 위치인 경우에는 에피소드가 종료되고 패널티  $R_f$  또는  $R_r$ 를 에이전트에게 준다. 따라서 에이전트는 어떠한 규칙도 모르는 상태에서 가장 먼저 트랜스포터를 이용한 블록 운반 작업 규칙을 학습하게 된다.

#### 3.1.1 블록 운반 작업 스케줄

블록 운반 작업 스케줄은 블록 적치 및 반출 작업을 포함하는 리스트이다. 시뮬레이터는 매 에피소드에서 가장 먼저 블록 운반 작업 스케줄을 생성한다. 즉, 매 에피소드 각기 다른 스케줄을 기반으로 시뮬레이션이 수행된다. 블록 운반 작업 스케줄은 블록 아이디, 블록의 적치 날짜, 반출 날짜, 적치기간 정보 제공 유무로 구성된다. 적치날짜는 연/월/일/시의 형태이며

반출날짜는 연/월/일의 형태이다. 적치날짜는 시각까지 생성하는데 반해 반출날짜의 경우 일까지만 생성하는 이유는 실제 블록 반출 작업 시 정확한 시간이 고려되지 않는다. 특정 날짜만으로 작업이 요청된 후 블록 운반 작업자들의 노하우에 의존하여 반출 작업 시각이 결정되기 때문이다.

블록 운반 작업 스케줄은 그림 4.3와 같이 블록의 적치 작업 발생 주기 ( $T$ ), 블록 적치장 포화율( $U$ ), 에이전트에게 제공되는 적치 기간 정보 비율 ( $P$ )을 이용하여 만든다. 블록 운반 작업 스케줄은 적치 작업과 적치 작업 사이의 평균 시간을 나타내는  $T$ 와 한 블록의 평균 적치 기간을 나타내는  $S$ 가 포아송 과정(Poisson process)을 따르는 M/M/1 대기행렬모델에 따라 생성된다. 적치장 포화율을 쉽게 조절할 수 있도록  $U$ 를 입력으로 받으며, M/M/1 모델에 따라  $U = \frac{T}{S}$ 이므로  $S = \frac{T}{U}$  공식을 이용하여  $S$ 를 구한다. 그림 3.2는  $T = 2$ ,  $U = 70\%$ ,  $P = 50\%$ 인 블록 운반 작업 스케줄 예시이다.

적치기간 정보 제공 유무는 참/거짓의 값을 가지며, 블록 적치 이후 에이전트에게 블록의 반출 작업까지 남은 적치 기간 정보를 제공하는것에 대한 여부를 나타낸다. 적치 기간 정보 제공 비율은  $P$ 를 통해 지정할 수 있다. 이 기능을 제공하는 이유는 실제 블록 적치장 내 적치되어 있는 블록은 반출 날짜를 미리 아는 경우도 있지만, 모르는 경우도 있기 때문이다. 따라서 에이전트에게 제공되는 적치 기간 정보의 비율을 조절하는 것은 각 조선소의 상황에 알맞는 시뮬레이션 환경을 구축하기 위해 필요한 요소이다.

0	2022-05-01 10:00:00	2022-05-05 00:00:00	True
1	2022-05-01 14:00:00	2022-05-02 00:00:00	False
2	2022-05-01 19:00:00	2022-05-02 00:00:00	False
3	2022-05-01 22:00:00	2022-05-02 00:00:00	False
4	2022-05-01 22:00:00	2022-05-06 00:00:00	False
5	2022-05-02 03:00:00	2022-05-03 00:00:00	False
6	2022-05-02 06:00:00	2022-05-03 00:00:00	False
7	2022-05-02 08:00:00	2022-05-05 00:00:00	False
8	2022-05-02 12:00:00	2022-05-03 00:00:00	True
9	2022-05-02 16:00:00	2022-05-03 00:00:00	True
10	2022-05-02 17:00:00	2022-05-05 00:00:00	False
11	2022-05-02 21:00:00	2022-05-03 00:00:00	True
12	2022-05-02 23:00:00	2022-05-05 00:00:00	True
13	2022-05-03 00:00:00	2022-05-04 00:00:00	False
14	2022-05-03 03:00:00	2022-05-06 00:00:00	True
15	2022-05-03 08:00:00	2022-05-05 00:00:00	True
16	2022-05-03 11:00:00	2022-05-04 00:00:00	False
17	2022-05-03 14:00:00	2022-05-04 00:00:00	False
18	2022-05-03 18:00:00	2022-05-04 00:00:00	True
19	2022-05-03 21:00:00	2022-05-04 00:00:00	True
20	2022-05-03 21:00:00	2022-05-06 00:00:00	False

그림 3.2: 블록 운반 작업 스케줄 예시

### 3.1.2 블록 적치 작업

블록 적치 작업은 블록 적치장에 새로운 블록을 배치하는 작업이다. 스케줄 내 모든 블록 적치 작업의 적치 위치는 에이전트의 행동에 의해 결정된다. 블록 적치 위치가 결정되면 적치 가능 여부를 판단하여 가능할 시, 해당 위치에 블록을 적치한다. 적치 가능 여부 판단은 출입구에서 적치 위치까지 트랜스포터 이동 경로 존재 유무로 결정한다. 적치 위치에 블록이 존재하거나 트랜스포터를 이용한 운반 가능 경로가 없을 경우 적치 작업은 실패한다. 적치 작업이 실패한 경우, 에이전트에게 적절하지 못한 결정에 대한 시그널을 주고 새로운 에피소드를 시작한다. 적치 작업을 성공한 경우, 현재 적치 작업 시각과 다음 적치 작업 시각 사이에 반출 작업의 유무를 확인한 뒤, 반출 작업이 있으면 반출 작업을 수행하고 반출 작업이 없으면

다음 적치 작업을 수행한다.

### 3.1.3 블록 반출 작업

블록 반출 작업은 블록 적치장에 적치되어 있는 블록을 공정 작업을 위해 적치장 출입구로 내보내는 작업이다. 반출 작업의 기한을 지키는 것은 굉장히 중요한 일이다. 따라서 반출 작업이 불가능한 경우 적치 작업보다 블록 재배치 작업을 먼저 수행한다. 블록 반출 작업은 블록의 위치에서 출입구까지 트랜스포터 이동 경로가 하나 이상 존재할 경우에 수행할 수 있다. 블록 반출 작업 가능시, 블록 적치장에서 블록을 제거한다. 블록 반출 작업 발생 기준은 적치 작업 시각 사이에 존재하는 반출 작업이다. 따라서 두개 이상의 블록 반출 작업이 존재할 수 있으며, 이러한 경우 반출 작업의 순서는 매우 중요하다. 반출 작업 순서에 따라서 재배치 작업 발생 횟수가 달라질 수 있다. 우리의 시뮬레이터는 블록과 출입구 사이의 거리를 이용하여 반출 작업 순서를 결정하며, 출입구에서 가까운 블록부터 반출 작업을 수행한다. 그 이유는 출입구에 가까이 적치된 블록일수록 트랜스포터의 이동 경로가 짧고 재배치 작업이 발생할 확률이 적으며, 멀리 적치된 블록 일수록 트랜스포터의 이동 경로가 길고 다른 반출 작업 대상 블록이 간접 블록이 될 확률이 높기 때문이다. 모든 반출 작업이 성공하면 다음 적치 작업을 수행한다. 그렇지 않은 경우, 반출 작업 경로 확보를 위해 블록 재배치

작업을 수행한다.

### 3.1.4 블록 재배치 작업

반출 작업 불가능 시, 블록 재배치 작업이 발생한다. 재배치 작업은 적 치장 내 적치되어 있는 블록을 새로운 위치로 이동시키는 작업이다. 블록 반출 경로 확보를 위해 다양한 블록 재배치 작업이 가능하며, 재배치 작업 대상 블록을 선정하는 것은 매우 중요하다. 우리 시뮬레이터는 반출 작업 대상 블록에서 출입구까지 간접 블록의 개수가 최소화되는 경로를 탐색하여 재배치 작업 대상 블록을 결정한다. 에이전트는 재배치 작업이 무엇인지 이해하고, 재배치 작업을 최소화해야 한다. 따라서 재배치 작업의 위치는 에이전트에 의해서 결정된다. 재배치 작업은 스케줄에 삽입되어 바로 다음 블록 적치 프로세스를 통해 수행된다. 우리는 재배치 작업 최소화하는 배치 전략 학습을 위하여 재배치 작업이 발생하는 경우 에이전트에게 음의 보상 신호인  $R_r$ 을 주었다.

## 3.2 강화학습 구성 요소

### 3.2.1 상태 (State)

에이전트는 환경으로부터 주어지는 상태를 이해하고 행동을 통해 상태를 변화시킨다. 블록 배치 환경에서 제공하는 상태는 블록 적치장이다. 블록 적치장은 블록들의 적치 위치, 블록의 남은 적치 기간 정보를 포함한다. 매 에피소드 시작시  $N \times M$ 의 블록 적치장은 -1로 초기화 된다. 블록 적치 작업 성공시, 블록의 적치 기간 정보 제공 여부에 따라 참인 경우 해당 위치는 블록의 남은 적치 기간으로 갱신되고, 거짓인 경우 -3으로 갱신된다. 블록 적치 작업 및 재배치 작업 실패 시, 에이전트에 의해 결정된 위치를 -2로 갱신한다. 따라서 에이전트는 -2가 포함된 상태를 받을 때마다 음의 보상을 함께 받게 되므로 블록 적치 작업 규칙 및 재배치 작업 규칙에 대해 학습하게된다. 매 스텝, 블록 적치장 내 블록들의 남은 적치 기간 정보는 현재 적치 작업 시작을 기준으로 갱신된다.

### 3.2.2 행동 (Action)

에이전트의 행동은 블록 적치 작업 및 재배치 작업 위치를 결정하는 것이다. 액션 공간(Action space)은 블록 적치장에서 트랜스포터 출입구가 포함된 행 및 열을 제외한 크기와 같다. 에이전트의 행동은 적치 실패 시 곧장 행동에 대한 보상을 받지만, 재배치 작업의 경우에는 일련의 행동들이

모여 지연된 보상을 발생시킨다. 에이전트가 옳바른 행동을 하기 위해서는 상태를 관찰하며, 미래에 발생할 수 있는 재배치 작업을 예상할 수 있어야 한다.

### 3.2.3 보상 (Reward)

보상은 강화학습을 통해 해결하고자 하는 문제를 정의하는 것이다. 에이전트의 최종 목표는 미래에 얻게될 누적 보상의 합이 최대화하는 것이다. 따라서 보상은 에이전트가 시행착오를 통해 최종 목표에 도달할 수 있도록 문제를 잘 정의하고, 빠르게 최종 목표를 달성할 수 있도록 설계하는 것이 중요하다. 블록 배치 환경의 최종 목표는 재배치 작업을 최소화하는 블록 배치 전략을 학습하는 것이다. 앞서 설명했듯, 이를 달성하기 위해서는 먼저 블록 배치 및 재배치 작업 규칙을 이해한 후에 재배치 작업을 최소화하는 블록 배치 전략을 학습해야한다.

블록 배치 환경은 그림 4.3과 같이 세 개의 보상함수  $R_f$ ,  $R_r$ ,  $R_c$ 를 제공한다.  $R_f$ 는 블록 적치 및 재배치 작업 규칙을 학습시키기 위한 것으로, 적치 및 재배치 작업 실패시 음의 보상을 부여한다.  $R_f$ 는 행동에 대해 곧장 발생하는 보상이므로 에이전트가 빠르게 블록 운반 규칙을 학습할 수 있다.  $R_r$ 은 재배치 작업 발생 시 음의 보상을 부여한다. 재배치 작업은 블록 운반 규칙을 이해하기 전인 학습 초기에는 거의 발생하지 않으며, 블록

적치장내에 블록이 충분히 쌓일 때 발생하게 된다. 따라서 인공신경망 기반의 에이전트가 학습 초기  $R_f$ 에 과적합되지 않고, 학습 중반 이 후  $R_r$ 을 통해 블록 재배치 작업 최소화 배치 전략을 학습할 수 있어야한다.

## 제 4 장 실험 및 성능 평가

본 장에서는 다양한 시나리오의 실험 및 분석을 통해 블록 배치 환경의 실용성 및 유용성을 증명한다. 우리는 추후 연구자 및 이용자가 새로운 아이디어를 적용하여 블록 배치 환경과 에이전트의 학습 성능 개선을 희망하며, 새로운 아이디어의 성능 평가를 위해 기준 성능(baseline performance)을 제공한다. 블록 배치 문제는 현실의 요소를 많이 반영할 수록 최적의 블록 배치 전략을 수립하는 것이 어려워진다. 따라서 우리는 각 조선소의 상황에 적합한 블록 배치 전략을 학습하기 위해 고려되어야하는 요소를 정의하고, 각 요소가 학습에 어떤 영향을 끼치는지 실험을 통해 보인다.

표 4.1은 본 장에서 실험할 시나리오들의 실험 조건을 정리한 내용이다. 블록 배치 전략 수립에 영향을 끼치는 요소를 에이전트, 블록 적치장의 모양, 블록 적치장 이용률, 블록 적치 기간 정보, 트랜스포터 출입구의 수, 보상 신호 체계로 정의하고, 각 요소가 블록 배치 전략 학습에 끼치는 영향을 실험을 통해 분석한다. 위 요소들의 조합은 수 없이 많이 존재할 수 있다. 우리는 각 요소가 끼치는 영향을 관찰하기 위해 하나의 시나리오는 하나의 요소만 변화시키며, 나머지 요소는 고정된 환경에서 실험을 진행한다.

강화학습의 중요한 특징 중 하나는 문제와 관련된 어떠한 규칙도 에이전트에게 미리 알려주지 않는다는 것이다. 에이전트가 최적의 블록 배치

그림 4.1: Experiment scenario list and conditions

	Agent	stockyard utilization	block storage information	reward function [ $R_f, R_r, R_c$ ]
Scenario 1	DQN, REINFORCE, PPO	50%	100%	[-5, -1, +1]
Scenario 3	PPO	50%, 70%, 90%	100%	[-5, -1, +1]
Scenario 4	PPO	70%	0%, 50%, 100%	[-5, -1, +1]
Scenario 5	PPO	70%	100%	[-5, -1, +1], [-1, -5, +1], [-1, -1, +5]

전략을 수립하기 위해서는 블록 적치 및 반출 작업에 대한 규칙을 이해한 후, 재배치 작업을 최소화할 수 있는 블록 배치 전략 탐색해야한다. 블록 운반 작업 규칙을 학습을 통해 이해해야지만 블록 적치장에 블록이 쌓이게 된다. 블록 적치장에 많은 블록이 있을수록 반출 작업 시 재배치 작업이 빈번히 발생하게 되어 재배치 작업 최소화 전략을 빠르게 학습할 수 있게 된다. 앞서 정의한 각 요소에 따라 블록 배치 전략 학습 난이도가 달라질 수 있다. 각 요소의 조합에 따라 최적의 학습 횟수는 상이하며, 우리의 목표는 최적의 학습 횟수를 찾는것이 아닌, 각 요소가 학습에 끼치는 영향을 관찰하기 위한 것임으로 모든 실험은 100,000회의 학습을 수행했다.

우리는 강화학습을 통해 수립된 블록 배치 전략의 성능 평가를 위한 지표로 10,000번의 에피소드를 진행하여 얻어진 평균 에피소드 완료 횟수와 평균 재배치 작업 횟수를 이용한다. 대부분의 기존 연구는 알고리즘의 성능 평가를 위해 재배치 작업 횟수를 이용였다. 하지만 재배치 작업 횟수만 고려할 경우, 주어진 블록 운반 작업 스케줄의 블록 적치 및 반출 작업을 끝까지 수행해내지 못했을 때 더욱 좋은 평가를 얻게된다. 블록 적치장 이용률이 높아 많은 블록이 적치되어 있는 경우, 좋지 않은 블록 배치 전략은 스케줄 내 블록 적치 및 반출 작업을 더이상 수행할 수 없게되어 에피소드가 종료된다. 반대로 좋은 블록 배치 전략은 재배치 작업이 조금 더 발생하더라도 스케줄 내 모든 작업을 완료해 낼 것이다. 재배치 작업 횟수만을

성능 지표로 이용할 경우 이러한 부분을 반영할 수 없다. 따라서 우리는 재배치 작업 횟수 뿐만 아니라 에피소드 완료 횟수를 함께 성능 지표로 이용한다.

## 4.1 강화학습 알고리즘별 학습 성능 분석

에이전트는 블록 배치 환경안에서 블록 적치장을 관찰하고 블록 재배치 작업이 최소화되도록 블록을 배치해야한다. 2.2장의 내용과 같이 강화학습 에이전트가 학습하는 방법으로 가치 기반 방법, 정책 기반 방법, 액터-크리틱 방법이 있다. 우리는 각 방법의 대표적인 알고리즘을 이용하여 블록 배치 전략 학습에 최적인 학습 방법을 관찰한다. 가치 기반 방법의 알고리즘으로 DQN, 정책 기반 방법의 알고리즘으로 REINFORCE, 액터-크리틱 방법의 알고리즘으로 PPO를 이용하여 각 알고리즘의 학습 성능을 비교 한다. 동등한 비교를 위해 모든 알고리즘은 동일한 하이퍼파라미터 값을 사용했다.

그림 4.1은 표 4.1 시나리오 1의 조건에서 DQN, PPO, REINFORCE 알고리즘을 이용하여 각 100,000만번의 학습 후, 학습이 완료된 모델을 이용하여 10,000만번의 시뮬레이션동안 성공적으로 스케줄 내 모든 작업을 완료한 에피소드의 수를 나타낸다. 실험의 수치는 다섯번의 시뮬레이션을 통해 얻은 값의 평균을 이용하였으며, 오차막대는 95% 신뢰구간을 나타낸다. 10,000번의 에피소드동안 PPO 알고리즘은 평균 9094회 성공하여 가장 높은 성능을 보였으며, DQN 알고리즘이 평균 8930회 성공하였고, REINFORCE 알고리즘이 평균 4920회 성공하여 가장 낮은 성능을 보였다. 에피소드 완료 횟수 지표에서는 PPO 알고리즘의 블록 배치 성능이 가장

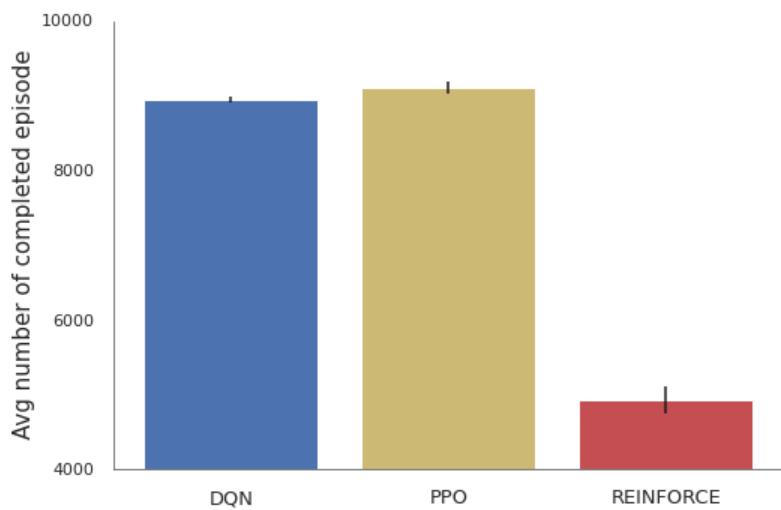


그림 4.1: 에이전트에 따라 10,000번의 시뮬레이션동안 스케줄 내 블록 운반 작업을 모두 완료한 에피소드 수. 오차막대는 95% 신뢰구간을 나타낸다.

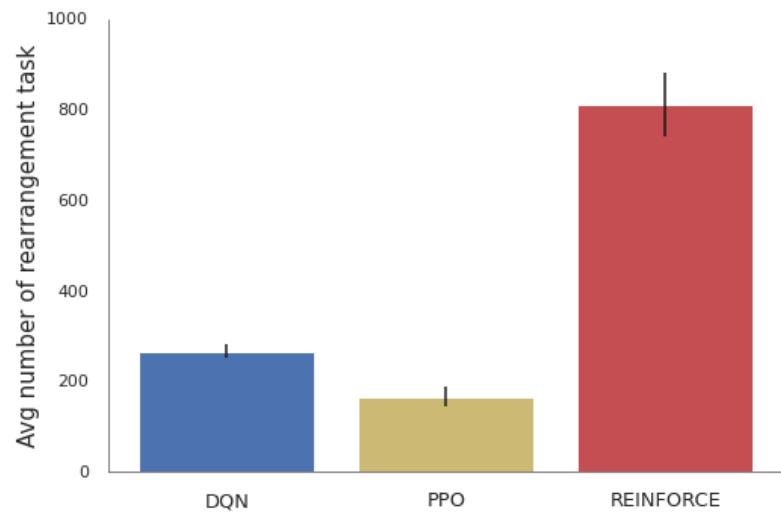


그림 4.2: 에이전트에 따라 10,000번의 시뮬레이션동안 발생한 재배치 작업 횟수

뛰어났다.

그림 4.2는 10,000번의 시뮬레이션동안 발생한 평균 재배치 작업 수를 나타낸다. PPO 알고리즘은 평균 164회의 가장 적은 재배치 작업이 발생했으며, DQN 알고리즘은 2번째로 적은 평균 265회의 재배치 작업이 발생했다. REINFORCE 알고리즘은 평균 809회의 가장 많은 재배치 작업이

발생했다. 결과적으로, 재배치 작업 발생 횟수 지표에서도 PPO 알고리즘이 가장 뛰어난 블록 배치 전략을 수립한 것을 알 수 있다. 따라서 위 두 실험을 통해 두 지표 모두에서 가장 뛰어난 블록 배치 성능을 보인 알고리즘은 액터-크리틱 방법의 PPO 알고리즘이며, PPO 알고리즘을 이용하여 이후의 실험들에 대한 학습 성능을 관찰한다.

그림 4.3은 PPO 알고리즘의 블록 배치 정책을 관찰한 것이다. 왼쪽/위에서 시작하여 적치작업을 수행할 때마다 블록 적치장의 상태를 나타낸다. 19번째 블록을 적치할 때까지 어떤 블록을 반출해도 재배치 작업이 발생하지 않도록 블록을 배치하는 것을 알 수 있다. 블록이 쌓일 수록 적치장내에 임의의 길을 유지하며 블록을 배치함으로써 재배치 작업을 최소화하는 것을 관찰할 수 있다.

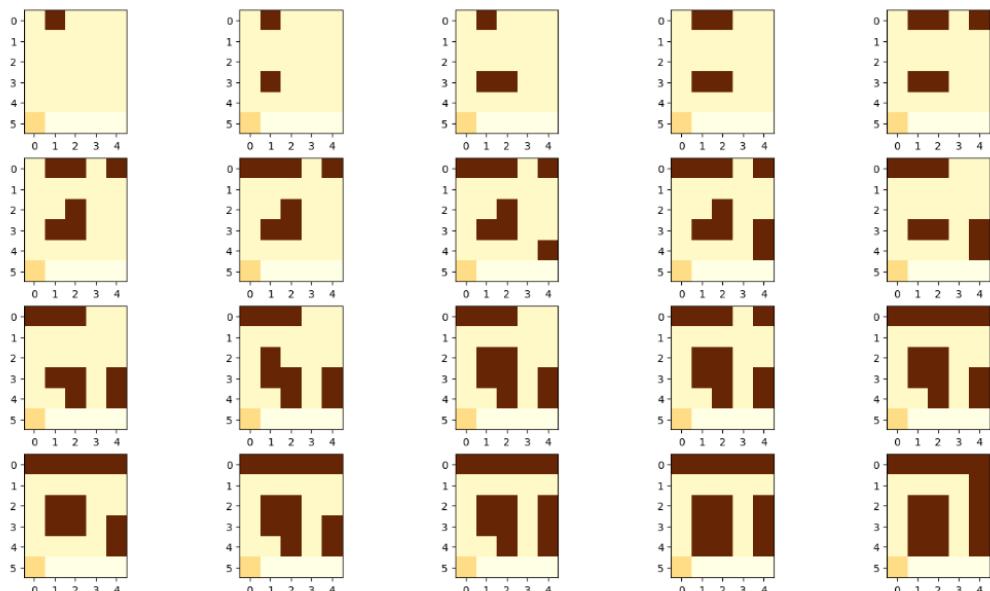


그림 4.3: PPO 알고리즘기반 블록 배치 예제

## 4.2 블록 적치장 포화율에 따른 학습 성능 분석

블록 적치장 포화율은 전체 적치장 면적 중 블록이 적치되어 사용되고 있는 공간의 비율을 의미한다. 포화율이 높아질수록 블록 반출 작업 시 트랜스포터 이동 경로 확보가 어려워 재배치 작업이 많이 발생하게 된다. 즉, 적치장 포화율은 재배치 작업 발생에 큰 영향을 끼치며, 포화율에 따라 블록 배치 문제의 난이도가 달라진다. 따라서 현실의 블록 적치장에 적합한 블록 배치 전략을 학습하기 위해서는 포화율이 반드시 고려되어야 한다.

3.1.1장에서 설명했듯이, 블록 운반 작업 스케줄은 M/M/1 대기행렬모델에 따라 생성되며, 적치 작업 발생 시각을 조절하여 블록 적치장 포화율을 조절한다.

그림 4.4는 길이 20인 블록 운반 작업 스케줄을 이용하여 평균 적치장

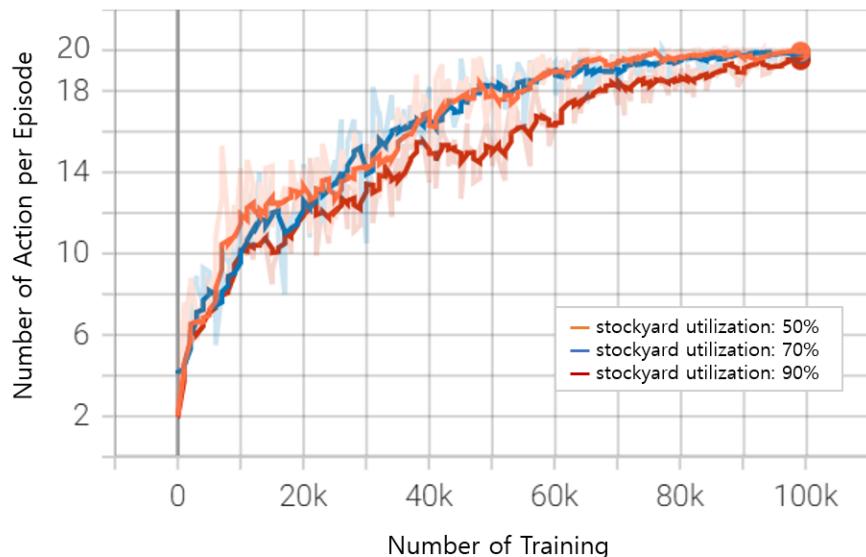


그림 4.4: 블록 적치장 포화율에 따른 학습 곡선

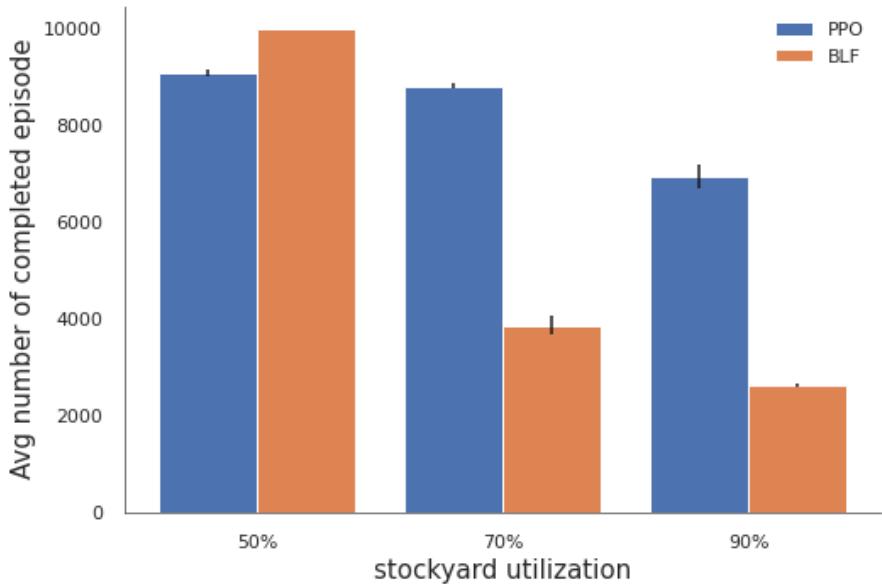


그림 4.5: 적치장 포화율에 따라 10,000번의 시뮬레이션동안 스케줄 내 블록 운반 작업을 모두 완료한 에피소드 수

포화율 별 학습 횟수에 따른 학습 곡선을 나타낸다. 포화율이 50%와 70%인 경우, 학습 곡선이 유사하여 100,000번의 학습만으로 에피소드당 블록 적치 작업이 20에 수렴하는 것을 알 수 있다. 포화율이 90%일 때, 가장 학습 속도가 느리고 100,000번의 학습으로 에피소드당 블록 적치 작업 횟수가 20에 수렴하지 못하는 것을 할 수 있다. 따라서 70%의 포화율까지는 에이전트가 최적의 블록 배치 정책을 학습하기 어렵지 않지만, 90%인 경우에는 다소 어려워진다는 것을 알 수 있다.

그림 4.5은 표 4.1 시나리오 3의 조건에서 100,000번의 학습을 완료한 PPO 알고리즘과 BLF를 이용하여 10,000만번의 시뮬레이션동안 성공적으로 스케줄 내 모든 작업을 완료한 에피소드의 수를 나타낸다. 포화율이 50%일 때, BLF 알고리즘은 모든 에피소드를 완료했지만, PPO 알고리즘은

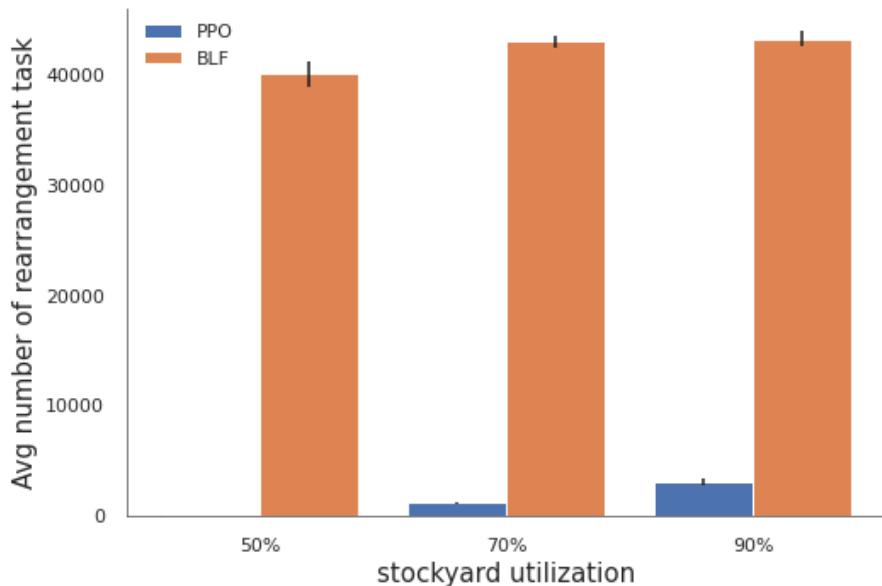


그림 4.6: 블록 적치장 포화율 별 재배치 작업 발생 횟수

9094회 완료했다. PPO 알고리즘이 모든 에피소드를 완료하지 못한 이유는 100,000번의 학습 동안 학습하지 못한 예외 케이스가 테스트동안 존재할 수 있기 때문이다. 적치장 포화율이 증가할 수록 각 알고리즘의 에피소드 완료 횟수는 줄어드는 것을 알 수 있으며, 평균 적치장 포화율이 70% 이상 일 때, PPO 알고리즘이 BLF 알고리즘보다 두배 이상 에피소드를 완료하였다. 따라서 적치장 포화율이 높을수록 PPO 알고리즘의 블록 배치 성능이 BLF 알고리즘보다 뛰어나다.

그림 4.6은 PPO알고리즘과 BLF 알고리즘이 10,000번의 시뮬레이션동안 발생시킨 블록 재배치 작업 횟수를 나타낸다. 블록 적치장 포화율에 관계없이 BLF 알고리즘의 블록 재배치 작업이 압도적으로 많이 발생했다. 그 이유는 BLF 알고리즘은 블록의 적치 및 재배치 작업 위치는 미래에 발생할

수 있는 재배치 작업을 고려하지 않는다. 하지만 PPO 알고리즘은 재배치 작업이 발생할 때마다 페널티를 받으며 학습하기 때문에 누적 보상의 합을 최대화하기 위해서 미래에 발생할 재배치 작업을 고려하여 블록 배치를 수행한다. 따라서 재배치 작업 최소화의 측면에서 강화학습기반 접근 방법이 월등히 뛰어난 블록 배치 성능을 보인다. PPO 알고리즘은 각 포화율에 따라 164, 1179, 3051회의 재배치 작업을 수행하였으며, 적치장 포화율이 증가할수록 블록 배치 문제의 난이도가 증가하기 때문에 재배치 작업이 많이 발생하는 것을 알 수 있다.

### 4.3 블록 적치 기간 정보에 따른 학습 성능 분석

블록 적치장에 대기되는 블록은 선박 및 공정에 따라 수 많은 종류가 존재한다. 따라서 블록 적치장에서 적치된 블록이 얼마나 오래 대기하는지에 관한 적치 기간 정보는 알 수도 있고 모를 수도 있다. 기존의 블록 배치 문제 관련연구는 이러한 요인을 고려하지 않기 때문에, 다소 실용성이 떨어지는 측면이 있다. 우리의 블록 배치 환경은 블록 적치 기간 정보 제공 비율을 반영하여 학습할 수 있다. 이 장에서는 모든 블록의 적치 기간 정보를 모를 때, 50%의 블록은 적치 기간 정보를 알고, 50%는 모를 때, 모든 블록의 적치 기간 정보를 알 때, 3가지 경우에 대해서 각 상황이 강화학습기반 블록 배치 전략 학습에 미치는 영향을 실험을 통해 관찰한다.

그림 4.7은 블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드의 수를 나타낸다. 에이전트가 모든 블록의 적치 기간 정보를 관찰 할 수 있을때 가장 많이 에피소드를 완료하였고, 50%의 블록 적치 기간 정보를 관찰할 때 에피소드 완료 횟수가 가장 적다.

실험 결과를 통해 알 수 있는 사실은 모든 블록의 적치 기간을 제공할 때, 에이전트는 블록 적치장의 숫자가 남은 적치 기간 정보라는 것을 이해할 수 있다는 것이다. 또한 블록의 적치 기간 정보를 전혀 제공하지 않을 때에도 나쁘지 않은 블록 배치 전략을 학습해낸다는 것을 알 수 있다. 가장 흥미

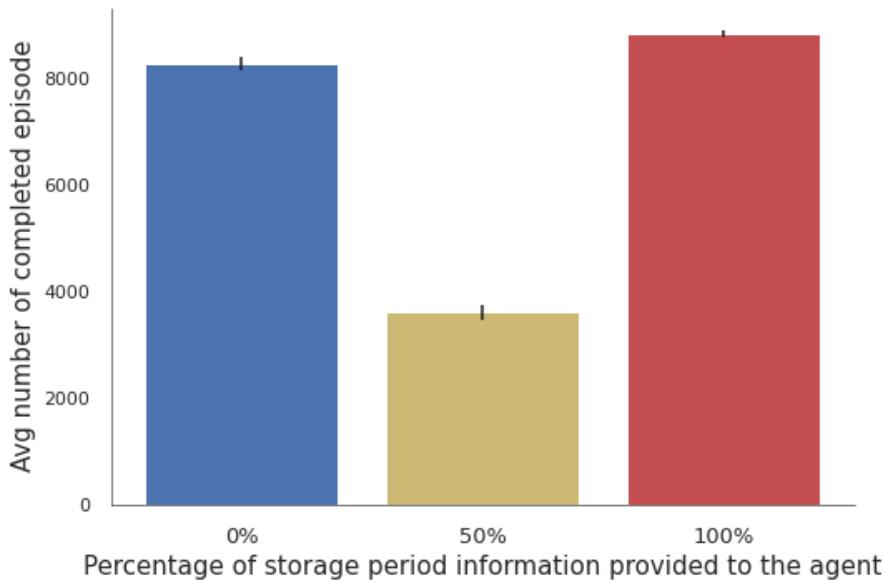


그림 4.7: 블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드 수

로운 점은 50%의 블록 적치 기간 정보를 제공할 때, 가장 안좋은 성능을 보이는데, 이는 에이전트가 이해해야할 정보의 양이 증가하기 때문이다. 모든 블록의 적치 기간 정보를 제공하면 에이전트는 상태를 관찰하며, 블록 적치장 내 숫자가 남은 블록의 적치 기간이라는 것을 학습한다. 블록의 적치 기간 정보를 제공하지 않으면 에이전트는 블록 적치장 내 숫자가 블록의 존재 여부라는 것을 학습한다. 하지만 50%는 적치 기간 정보를 제공하고, 50%는 고정된 값을 제공하면 에이전트는 블록 적치장 내 상태를 관찰하며 위 두가지 정보를 모두 이해해야한다. 따라서 학습해야하는 정보의 양이 증가하게 되며, 0%나 100%의 적치 기간 정보를 제공할 때와 달리 100,000회의 학습으로는 학습 횟수가 부족하여 환경을 관찰하며 상태를 정확히 이해하지 못했기 때문에 가장 낮은 성능을 보였다.

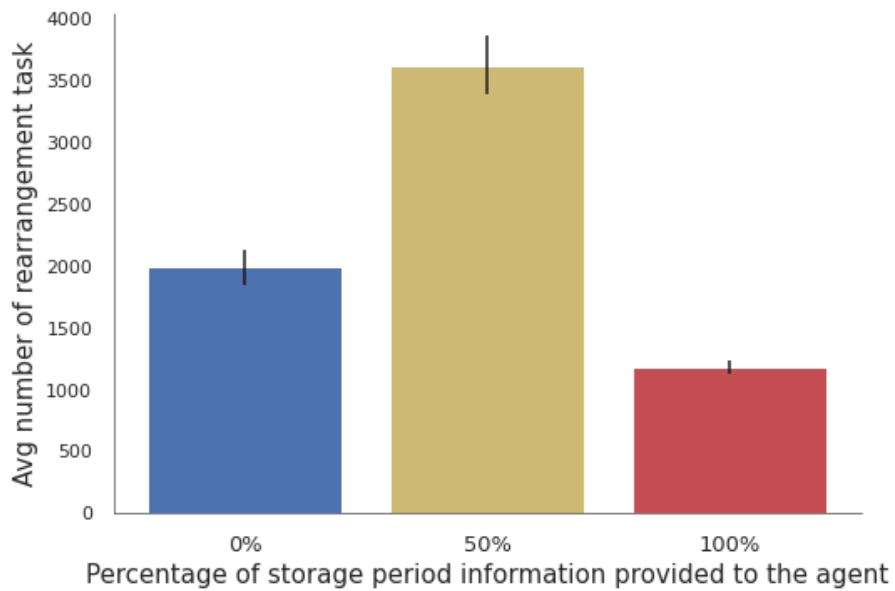


그림 4.8: 블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 발생한 재배치 작업 횟수

그림 4.8은 블록 적치 기간 정보 제공 비율에 따라 10,000만번의 시뮬레이션동안 발생한 재배치 작업 횟수를 나타낸다. 모든 블록의 적치 기간 정보를 제공할 때, 재배치 작업이 가장 적게 발생하며, 50%만 제공할 때, 재배치 작업이 가장 많이 발생한다. 에이전트는 모든 블록의 적치 기간 정보를 알 때, 이를 이용하여 재배치 작업을 최소화할 수 있다는 것을 알 수 있다. 50%의 적치 기간 정보를 관찰할 때 가장 많은 블록 재배치 작업이 발생한 이유는 앞의 분석 내용과 같이 학습 횟수의 부족으로 인해 상태를 정확히 이해하지 못했기 때문이다.

#### 4.4 보상 신호 조합에 따른 학습 성능 분석

3.2.3장에서 설명했듯이, 블록 배치 환경은 세개의 보상 함수를 제공한다. 최종 목표를 빠르게 달성하기 위해 에이전트에게 제공되는 보상 신호를 적절히 조절해야한다. 각 보상 함수는 설정 파일을 통해 쉽게 조절 가능하며, 이 장에서는 각 보상함수에서 발생하는 보상 신호의 크기가 학습 속도에 미치는 영향을 분석한다. 그림 4.9은 각 보상 함수에서 발생하는 보상 크기의 조합에 따른 학습 속도를 나타낸것이다. 각 보상 함수가 미치는 영향을 관찰하기 위해 하나의 보상 함수의 크기는 5 나머지 보상 함수의 크기는 1로 설정하였다.  $R_f$ 에 대해 -5의 보상을 주었을 때, 가장 빠르게 블록 운반 작업 규칙을 학습하여 20에 수렴하였고,  $R_r$ 에 대해 -5의 보상을 주었을 때, 학습 속도는 다소 느리지만 100,000회의 학습동안 20에 수렴하

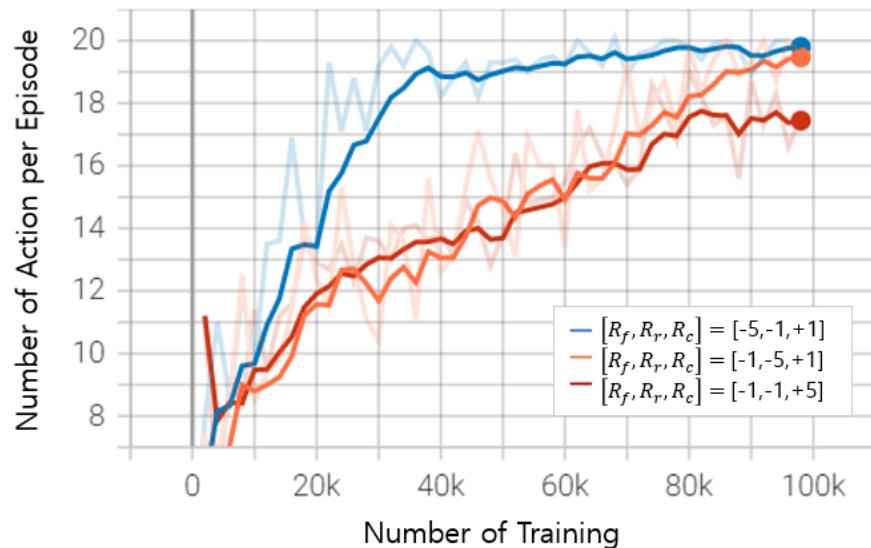


그림 4.9: 블록 적치장 포화율에 따른 학습 곡선

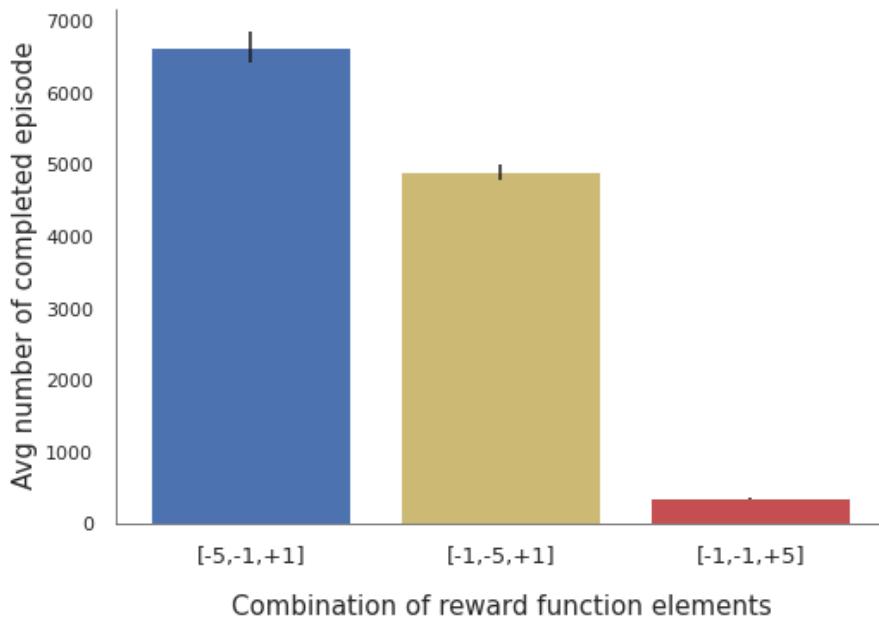


그림 4.10: 보상 신호 조합에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드 수

였다.  $R_c$ 에 대해 +5의 보상을 주었을 때는 같은 학습이 끝날때까지 20에 수렴하지 못했다. 시나리오5의 실험을 통해 각 보상 함수의 크기가 학습 속도에 영향을 미친다는 것을 알 수 있으며, 추후 연구자들은 블록 배치 환경의 설정 파일을 통해 쉽게 새로운 아이디어를 적용할 수 있다.

그림 4.10는 보상 크기의 조합에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드의 수를 나타내고, 그림 4.11는 보상 크기의 조합에 따라 10,000만번의 시뮬레이션동안 성공적으로 완료한 에피소드의 수를 나타낸다. 보상 함수 조합이  $R_f, R_r, R_c$ 가 -5,-1,+1일 때, 가장 많은 에피소드를 완료하였다.  $R_f$ 는 블록 적치 및 재배치 작업에 실패할 시 발생 하기 때문에  $R_f$ 가 클 때, 빠르게 블록 적치 및 재배치 작업에 대한 규칙을 학습하는 것을 알 수 있다. 보상 함수 조합이  $R_f, R_r, R_c$ 가 -1,-5,+1일 때,

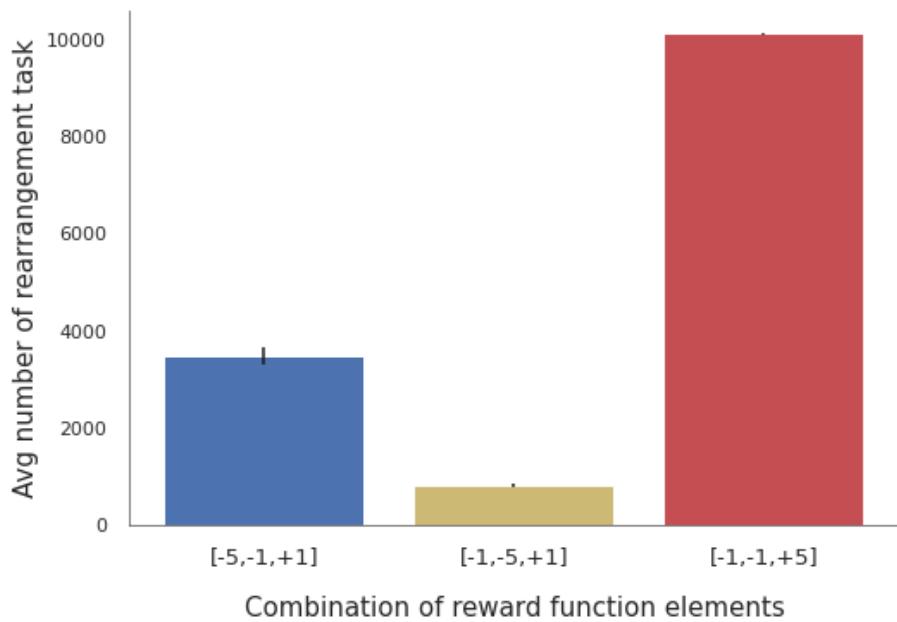


그림 4.11: 보상 신호 조합에 따라 10,000만번의 시뮬레이션동안 발생한 재배치 작업 횟수

에피소드 완료 횟수는 상대적으로 적지만 재배치 작업 횟수는 가장 적은 것을 알 수 있다.  $R_r$ 은 재배치 작업 발생 시 에이전트에게 주어지기 때문에  $R_r$ 이 클 때, 재배치 작업이 빠르게 최소화된다는 것을 알 수 있다. 마지막으로 보상 함수 조합이  $-1, -1, +5$ 일 때 두 지표 모두에서 가장 안좋은 블록 배치 성능을 보였다.  $R_c$ 는 다른 보상 함수에 비해 에피소드 당 한번 발생하며, 더욱이  $+5$ 는 굉장히 드물게 발생하기 때문에 에이전트가 빠르게 학습하지 못하였다.

## 제 5 장 결론

본 논문은 블록 적치장의 효율적인 운용을 위한 블록 배치 환경을 제안 한다. 블록 배치 환경은 강화학습을 이용하여 재배치 작업을 최소화하는 블록 배치 전략을 학습할 수 있고, 휴리스틱 블록 배치 알고리즘의 성능을 평가할 수 있다. 또한 각 조선소 및 블록 적치장의 특성에 맞는 블록 배치 전략을 학습할 수 있도록 쉽게 커스터마이징할 수 있다. 우리는 블록 배치 성능에 영향을 끼치는 요소들을 정의하고, 실험을 통해 각 요소가 끼치는 영향을 분석하였다. 동일한 조건에서 실험을 진행하기 위해 학습 횟수, 인공신경망의 하이퍼파라미터등을 동일하게 설정하였다. 따라서 추후 연구자들은 각 요소에 따른 최적의 학습 조건을 탐색할 수 있다. 우리는 블록 배치 환경에 새로운 아이디어가 적용되어 성능과 실용성 측면에서 개선되어 나가길 기대한다.

## 참고 문헌

- [1] C. Park, J. Seo, J. Kim, S. Lee, T.-H. Baek S.-K. Min (2007) Assembly block storage location assignment at a shipyard: a case of Hyundai Heavy Industries, Production Planning Control, 18:3, 180-189, DOI: 10.1080/09537280601009039
- [2] Changkyu Park, Junyong Seo (2009) Assembly block storage location assignment problem: revisited, Production Planning Control, 20:3, 216-226, DOI: 10.1080/09537280902803056
- [3] Changkyu Park, Junyong Seo, Comparing heuristic algorithms of the planar storage location assignment problem, Transportation Research Part E: Logistics and Transportation Review, Volume 46, Issue 1, 2010, Pages 171-185, ISSN 1366-5545, <https://doi.org/10.1016/j.tre.2009.07.004>.
- [4] Jong Gye Shin, Oh Heung Kwon Cheolho Ryu (2008) Heuristic and metaheuristic spatial planning of assembly blocks with process schedules in an assembly shop using differential evolution, Production Planning Control, 19:6, 605-615, DOI: 10.1080/09537280802474941

- [5] Ningrong Tao, Zuhua Jiang Shipeng Qu (2013) Assembly block location and sequencing for flat transporters in a planar storage yard of shipyards, International Journal of Production Research, 51:14, 4289-4301, DOI: 10.1080/00207543.2013.774477
- [6] Jeong, Y., S. Ju, H. Shen, D. Lee, J. Shin, and C. Ryu. 2018. “An Analysis of Shipyard Spatial Arrangement Planning Problems and a Spatial Arrangement Algorithm Considering Free Space and Unplaced Block.” International Journal of Advanced Manufacturing Technology 95: 4307–4325. doi: 10.1007/s00170-017-1525-1
- [7] Byeongseop Kim, Yongkuk Jeong Jong Gye Shin (2020) Spatial arrangement using deep reinforcement learning to minimise rearrangement in ship block stockyards, International Journal of Production Research, 58:16, 5062-5076, DOI: 10.1080/00207543.2020.1748247
- [8] Sutton, R., and A. Barto. 2012. A Reinforcement Learning: An Introduction. 2nd ed. Cambridge, MA: MIT Press.
- [9] Mnih, V. et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

- [10] Rumelhart, D., Hinton, G. Williams, R. Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986).  
<https://doi.org/10.1038/323533a0>
- [11] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99). MIT Press, Cambridge, MA, USA, 1057–1063.
- [12] Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. “Asynchronous Methods for Deep Reinforcement Learning.” *Proceedings of Machine Learning Research* 48: 1928–1937.
- [13] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*
- [14] P. Wagner. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. *Advances in Neural Information Processing Systems*, 24, 2011.

# Design and Evaluation of Reinforcement Learning Environment for Efficient Operation of Block Stockyard in Shipyard

**Guk-Cheol Choi**

Department of Information Convergence Engineering  
The Graduate School  
Pusan National University

## Abstract

Ship built in shipyards are divided into multiple blocks. Each block is completed through various processes, and it is very difficult to accurately meet the process schedule of all blocks because numerous blocks are processed at the same time. Therefore, there are frequent situations in which the block cannot immediately proceed with the next process and has to wait. The temporary waiting place of the block is called a block stack, and various block transport operations occur here. Block transportation takes a lot of time and money, so unnecessary block relocation work in the landfill causes a great loss. Therefore, placing blocks to minimize block relocation tasks is the most important factor for efficient block loading operation. This study proposes a block placement environment, a reinforcement learning environment suitable for learning block placement strategies for minimizing relocation tasks. The block placement environment provides a simulation of block transport operations occurring in the storage area, and the reinforcement learning agent interacts with the block placement environment and learns block placement strategies through trial and error. We define factors to consider for learning block placement strategies suitable for each shipyard, and experiments and analyses on various scenarios show that the block placement environment is a suitable environment for learning block placement strategies.