JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 29 No. 1, pp. 11-19, January 2024
https://doi.org/10.9708/jksci.2024.29.01.011

Cooperative Multi-agent Reinforcement Learning on Sparse Reward Battlefield Environment using QMIX and RND in Ray RLlib

Minkyoung Kim*

*Engineer, Infra Technology R&D Center, Hanwha Systems, Seongnam, Korea

[Abstract]

Multi-agent systems can be utilized in various real-world cooperative environments such as battlefield engagements and unmanned transport vehicles. In the context of battlefield engagements, where dense reward design faces challenges due to limited domain knowledge, it is crucial to consider situations that are learned through explicit sparse rewards. This paper explores the collaborative potential among allied agents in a battlefield scenario. Utilizing the Multi-Robot Warehouse Environment(RWARE) as a sparse reward environment, we define analogous problems and establish evaluation criteria. Constructing a learning environment with the QMIX algorithm from the reinforcement learning library Ray RLlib, we enhance the Agent Network of QMIX and integrate Random Network Distillation(RND). This enables the extraction of patterns and temporal features from partial observations of agents, confirming the potential for improving the acquisition of sparse reward experiences through intrinsic rewards.

▸ **Key words**: Multi-agent Reinforcement Learning, Sparse Reward, Cooperative Battlefield Engagement, Ray RLlib, QMIX, RND

[요 약]

멀티에이전트는 전장 교전 상황, 무인 운송 차량 등 다양한 실제 협동 환경에 사용될 수 있다. 전장 교전 상황에서는 도메인 정보의 제한으로 즉각적인 보상(Dense Reward) 설계의 어려움이 있어 명백한 희소 보상(Sparse Reward)으로 학습되는 상황을 고려해야 한다. 본 논문에서는 전장 교전 상황에서의 아군 에이전트 간 협업 가능성을 확인하며, 희소 보상 환경인 Multi-Robot Warehouse Environment(RWARE)를 활용하여 유사한 문제와 평가 기준을 정의하고, 강화학습 라이브러리인 Ray RLlib의 QMIX 알고리즘을 사용하여 학습 환경을 구성한다. 정의한 문제에 대해 QMIX의 Agent Network를 개선하고 Random Network Distillation(RND)을 적용한다. 이를 통해 에이전트의 부분 관측값에 대한 패턴과 시간 특징을 추출하고, 에이전트의 내적 보상(Intrinsic Reward)을 통해 희소 보상 경험 획득 개선이 가능함을 실험을 통해 확인한다.

▸ **주제어**: 멀티에이전트 강화학습, 희소 보상, 전장 교전 협업, Ray RLlib, QMIX, RND

• First Author: Minkyoung Kim, Corresponding Author: Minkyoung Kim
*Minkyoung Kim (kmk0224@hanwha.com), Infra Technology R&D Center, Hanwha Systems
• Received: 2023. 11. 23, Revised: 2023. 12. 28, Accepted: 2023. 12. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

AlphaGo의 등장과 함께 싱글 에이전트 강화학습(Single-Agent Reinforcement Learning)에 대한 연구가 활발하게 진행되었다. 하지만, 현실에서는 많은 에이전트가 서로를 인식하며, 협동(Cooperative)이나 경쟁(Competitive)의 목적을 가지고 행동을 수행해야 할 필요가 있다. 멀티에이전트 강화학습(Multi-Agent Reinforcement Learning)은 다수의 에이전트가 존재하여, 각 에이전트가 환경과 상호작용하며 보상을 최대화하는 방향으로 학습한다. 목표를 수행하는 다수의 협동 로봇이나, 여러 유무인 개체가 서로 협동하여 전술을 수행하는 전장 교전 상황[1], 무인기 개체의 분산 협업 상황[2], 또 물류센터에서 동작하는 무인 운송 차량(AGV: Automated Guided Vehicle) 등을 예로 들 수 있다.

국방 분야에서는 AI를 여러 체계에 적용하려는 시도가 계속되고 있으며, 유무인 복합체계(Man-Unmanned Teaming)를 넘어 미래 전장 교전 상황에서는 아군의 특정 목표를 위해 각 에이전트가 협업을 수행하는 경우가 발생할 것으로 예측한다. 하지만 이러한 전장 교전 상황에서의 협업은 많은 도메인 지식이 필요한 부분이며, 그 부분은 즉각적인 보상(Dense Reward)으로 정의하기 어려운 부분이 있다[3]. 명확하게 정의할 수 있는 보상은 대표적으로 아군 전투 승리 또는 아군 무장 발사에 따른 적 개체 격추[4] 등으로 예상되며, 이는 희소 보상(Sparse Reward)으로 정의된다. 이 보상은 많은 연속적인 행동들의 결과로 얻게 되는 장기 보상(Long-term Reward)에 해당하며, 오랜 시간 동안 실행되는 전장 교전 상황에서 드물게 발생한다. 즉, 전장 교전 상황에서 희소 보상을 고려한 에이전트 간 협업 문제에 대한 연구가 필요하다.

본 논문에서는 희소 보상 환경인 Multi-Robot Warehouse Environment(RWARE)[5] 시뮬레이터 환경에서 전장 교전 상황과 유사한 상황을 정의한 후, 희소 보상 환경에서의 멀티에이전트 강화학습 QMIX[6] 알고리즘을 이용한 협업 연구를 진행한다. 이때, 에이전트들은 중앙집중형 훈련-분산형 실행(CTDE: Centralized Training with Decentralized Execution) 방법으로 진행하여, 학습(Training) 과정 중에는 모든 에이전트의 관측 정보를 이용하고, 실행(Execution) 과정에서는 각 에이전트 자신의 관측 정보를 이용한다. 즉, 실행 시에 에이전트 간 통신을 통한 정보 교류가 불가능한 상황에서 협동 상황을 가정하여 협업을 목표로 학습을 진행한다.

본 논문의 주요 목적 및 기여점은 다음과 같다:

1) 희소 보상이 획득되는 전장 교전 상황에서의 아군 에이전트 간 협업 진행 가능성을 확인한다. 이를 위해 희소 보상 문제를 정의하고 있는 RWARE를 이용하여 전장 도메인과 유사한 상황을 가정하여 진행한다.
2) 협업 환경에서의 희소 보상 문제를 Random Network Distillation(RND)[7]을 적용하여 도메인 지식에 독립적으로 희소 보상 획득 개선이 가능함을 확인한다.
3) 강화학습 라이브러리인 Ray RLlib을 사용하여, QMIX 알고리즘의 개선한 Agent Network와 RND가 구현을 통해 적용 가능함을 확인한다.

본 논문은 다음과 같이 구성된다. 2장에서는 다른 강화학습 멀티에이전트 환경과 RWARE 차이를 설명하며, QMIX 알고리즘과 가치기반 멀티에이전트 강화학습 알고리즘을 비교한다. 또한, 강화학습 라이브러리인 Ray RLlib[8]와 RND에 대해 다룬다. 3장에서는 희소 보상 환경인 RWARE 환경에서 전투 교전 상황과 유사한 문제를 정의한다. 또한, RWARE와 Ray RLlib의 QMIX 알고리즘과 RND를 사용한 학습 구조를 제안한다. 4장에서는 실험 환경과 평가 기준을 정의하고, 3장에서 제안한 학습 결과를 정의한 평가 기준에 따라 분석하며, 5장에서 결론을 진행한다.

II. Preliminaries

1. RWARE

Multi-Robot Warehouse Environment(RWARE)는 SMAC(StarCraft Multi-Agent Challenge)[9], LBF(Level-Based Foraging)[5]와 더불어 대표적인 멀티에이전트 강화학습 환경이다. 각 환경은 관측값의 획득 범위, 보상의 희소성 여부, 그리고 해결하고자 하는 주된 문제가 다르다[표1]. SMAC은 각 에이전트의 다양한 종족 특성에 따라 행동과 공격하는 목표물이 달라지는 환경의 멀티에이전트 문제를 다루며, 밀도 있는 보상(Dense Reward)을 받는다. LBF와 RWARE는 모두 보상이 희소한 경우를 다루기 때문에 탐험(Exploration)에 대해 고려하는 부분이 유사하지만, LBF는 목표물의 우선순위가 정해져 있고, 에이전트별로 목표물을 획득할 수 있는 조건이 상황에 따라 변화하기 때문에 협동 부분에 더 비중을 두고 있다.

Table 1. Compare properties of a multi-agent reinforcement learning environment

Environ-ment	Observability	Reward Sparsity	Main Difficulty
SMAC	Partial	Dense	Large action space
LBF	Partial/Full	Sparse	Coordination
RWARE	Partial	Sparse	Sparse reward

RWARE 시뮬레이터는 창고 환경에서 여러 개의 로봇 AGV 에이전트가 요청받은 선반을 목적지인 워크스테이션까지 배달하는 상황을 모사한다. 에이전트는 자신의 그리드 셀을 중심으로 제한적인 부분 관측값(Partial Observation)을 획득하여 동작을 수행할 수 있다. 에이전트는 요구되는 선반을 목적지까지 배달을 완료해야만 보상을 얻을 수 있기 때문에, 다른 강화학습 환경과 비교할 때 희소성 있는 보상을 얻게 된다[5]. 특히 에이전트가 보상을 얻기 위해서는 순차적인 행동 시퀀스를 올바르게 완료해야 하므로, 희소한 보상이 학습되기 어려운 환경 조건을 만든다. 이러한 환경 특성으로 본 논문에서는 RWARE 시뮬레이터를 사용한다.

2. QMIX

QMIX[6]는 멀티에이전트 강화학습 분야에서 사용하는 대표적인 오프-폴리시(Off-policy) 가치기반(Value-based) 알고리즘이며, 서로 협력하는 멀티에이전트 환경에 주로 사용한다. 협력적 가치기반 멀티에이전트는 서로 간의 행동에 대한 영향성을 반영하여 학습하기 위해 주로 전체 상태(Global State)와 공동 행동(Joint Action)을 반영한 중앙 행동-가치 함수(Centralized Action-Value Function)를 학습한다.

가장 간단한 형태의 중앙 행동-가치 함수는 에이전트 별로 행동-가치 함수를 만들어 학습하는 IQN(Independent Q-Learning)[10]이다, 이 알고리즘은 에이전트 간 상호작용을 반영하지 못한다는 한계점이 있으며, 비정상성 분포(Non-stationary)에 대한 불안정성(Instability)이 있다. VDN(Value Decomposition Networks)[11]은 중앙 행동-가치 함수가 에이전트별 행동-가치 함수의 합으로 구성되어, 학습 시 전체 상태와 같은 추가적인 정보를 이용하여 학습할 수 없는 한계점이 있다.

QMIX는 공동-가치 함수를 복잡한 비선형 방식의 인공신경망으로 계산하는 Mixing Network를 사용하여, 전체 상태와 공동 행동을 반영하여 학습 가능하다. VDN과 같은 선형 방식에서 Mixing Network 개념 적용을 위해 중앙 행동-가치 함수는 에이전트별 행동-가치 함수에 대한 단조 증가 함수여야 한다는 조건을 추가하여, 양수의 신경망 가중치를 가져야 하는 조건이 있다. RWARE는 요청된 선반을 옮긴다는 팀의 목표를 위해 에이전트들의 협동이 필요하므로 다중 에이전트의 협력적 수행을 위해 QMIX 알고리즘을 사용했으며, RWARE 시뮬레이터의 정의된 희소 보상은 양수의 신경망 가중치를 업데이트할 수 있다.

3. Ray RLlib

Ray RLlib[8]은 강화학습 라이브러리로 강화학습과 관련된 멀티에이전트 환경, 오프라인 데이터셋, 외부의 시뮬레이터와 연동하는 API 등을 제공한다. 특히 OpenAI Gym 환경을 포함한 Standard Environment 연동을 지원하여 학습 구성이 용이하다. 또한 TensorFlow와 PyTorch 기반의 A2C, PPO, DQN과 같은 26개의 싱글 에이전트 강화학습 알고리즘과 2개의 멀티에이전트 강화학습 알고리즘인 QMIX와 MADDPG를 지원한다. 2017년부터 총 72번의 버전 릴리즈가 진행되어 지속해서 업데이트되는 안정적인 강화학습 라이브러리 중 하나이다. OpenAI Gym 형태의 RWARE 시뮬레이터 구조와 Ray의 멀티에이전트 환경 API 제공 및 QMIX 알고리즘 지원으로 Ray RLlib를 사용하여 학습 환경을 구성한다.

4. Random Network Distillation

외적 보상(Extrinsic Reward)은 에이전트가 목표에 도달했을 때, 환경으로부터 받는 보상으로 직접적으로 에이전트의 목표와 연관 있다. 반면, 내적 보상(Intrinsic Reward)은 에이전트가 경험하지 못한 상태를 만나면 추가 보상을 부여하는 방법으로, 에이전트에게 내재적 동기를 부여하여 탐험을 유도한다. 희소 보상 환경에서는 에이전트가 보상을 얻기 어려운 환경이기 때문에, 탐험을 통한 경험 획득이 중요하다. 이에 RND는 희소 보상 환경에서 에이전트에게 추가적인 정보를 제공하여, 유용한 탐험을 유도하여 효율성을 향상한다.

RND는 동일한 구조의 두 개의 신경망을 사용하여, 예측 신경망(Prediction Network)과 무작위 신경망(Target Network)으로 구성된다. 두 신경망은 학습 초기에 무작위로 초기화된다. 무작위 신경망을 통해 예측 신경망을 학습시키며, 무작위 신경망 출력과 예측 신경망 출력의 오차를 통해 내적 보상을 정의한다. 이 오차는 에이전트가 경험하지 못한 상태일수록 커지며, 이 값을 통해 에이전트의 경험에 대한 중요도를 평가할 수 있다.

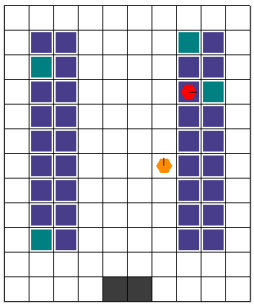

Fig. 1. Multi-Robot Warehouse 10×12 grid environment

III. The Proposed Scheme

희소 보상이 획득되는 전장 교전 상황에서의 아군 에이전트의 협업 진행 가능성을 확인하기 위해, 희소 보상 문제를 정의하고 있는 RWARE 환경을 통해 전장 교전 도메인과 유사한 상황을 가정한다. 전장 교전 상황은 2개의 아군 에이전트가 무작위로 생성되는 4개의 적 타겟 개체에 대하여 격추하는 상황을 가정한다. 이때, 실행 시 에이전트들의 탐지 범위는 제한되며, 서로 통신 채널을 통해 정보 공유가 불가능하다. 에이전트들은 탐지 범위 내 아군 개체의 정보, 적군 개체의 정보, 적군 타겟 개체의 정보를 획득할 수 있다. 에이전트는 기동과 무장 선택 및 발사의 행동을 수행한다. 에이전트가 적 타겟 개체에 대한 무장을 선택 및 발사하여, 격추에 성공하면 +1의 희소 보상을 얻게 되며, 1) 타겟 위치 탐색, 2) 무장 선택, 무장 발사 및 격추, 3) 회피를 순차적으로 반복 진행[4]하여 적 타겟 개체를 격추해 교전 상황에서 승리할 수 있다. 가정하는 전장 교전 상황에 대하여 아래 RWARE 환경에 대해 유사하게 정의하여 진행한다.

1. Define Problem

완전 협동 멀티에이전트 태스크(Fully Cooperative Multi-agent Task)는 Dec-POMDP(Decentralized Partially Observable Markov Decision Process)의 형태인 $G=\langle S, U, P, r, Z, O, \gamma\rangle$튜플로 정의 가능하다. 각 타임 스텝 t마다 RWARE 시뮬레이터의 10×12 그리드 셀 사이즈의 환경 상태 $s_t \in S$ [그림1]를 가정하며, 2개의 AGV 에이전트 $a \in \{1,...,n\} \equiv A$가 존재한다. 에이전트들의 목표는 요청된 선반을 목적지인 워크스테이션까지 배달하는 것이며, 이때 보라색으로 표기된 셀이 선반, 초록색으로 표기된 셀이 요청된 선반, 그리고 회색으로 표기된 셀이 목적지이다. 실행 시, 에이전트들은 관찰 함수 O로 자신이 탐지한 셀 영역에서만 부분 관측값 Z을 획득하여 $O: S \times A \rightarrow Z$로 표기된다. 에이전트 간 통신을 통한 정보 공유가 불가능함을 가정한다.

1.1 Actions

각 에이전트 a가 할 수 있는 행동 u_t^a은 총 5가지로 $u^a \in$ {No Op, Move Forward, Turn Left, Turn Right, Toggle Load} $\equiv U$ 정의되며, 5가지 동작을 통해 그리드 셀을 상하좌우로 이동한다. 에이전트는 Toggle Load 행동을 통해서 선반을 들거나 내릴 수 있다. 선반을 들어 운반하는 에이전트는 빨간색으로 표기되고, 선반을 들지 않은 에이전트는 노란색으로 표기된다. 이때 선반을 들어 빨간색으로 표기된 에이전트는 보라색과 초록색으로 표기된 선반 밑으로 이동 불가능하며, 비어있는 복도 공간에서 이동할 수 있다. 에이전트는 들고 있는 선반을 내 복도를 제외한 목적지 그리드 셀과 비어있는 선반 그리드 셀만 내려놓을 수 있다. 또한, 이동하고자 하는 셀에 다른 에이전트가 존재하면 이동 불가능하다.

1.2 Observations

각 에이전트의 부분 관측값 z는 에이전트가 위치한 셀을 정중앙에 둔 3 x 3 그리드 셀 범위까지 획득할 수 있다. 획득 정보로는 다른 에이전트의 존재 유무, 다른 에이전트의 방향 정보, 선반의 존재 유무, 요청된 선반의 존재 유무가 포함된다. 상태는 상태 전이 함수인 $P(s_{t+1}|s_t, u_t): S \times U \times S \rightarrow [0,1]$에 따라 전이된다.

1.3 Sparse Rewards

에피소드마다 총 4개의 요청되는 선반이 무작위로 생성되며, 에이전트가 요청된 선반을 목적지까지 배달하면 보상 r을 +1점 얻게 된다. 보상은 $r(s,u): S \times U \rightarrow R$ 로 표기되며, 감가율은 $\gamma \in [0,1]$에 따른다. 요청된 선반이 목적지에 배달되면 보라색으로 색상이 변경되며, 요청되는 선반이 새로 추가되어 4개의 요청 선반 개수가 유지된다. 즉, 에이전트가 보상을 얻기 위해서는 1) 무작위로 생성되는 요청 선반의 위치를 찾고, 2) 해당 선반을 가지고 목적지까지 배달하고, 3) 배달한 선반을 다시 비어있는 선반 셀에 내려놓아야 하는 순차적인 행동을 반복해야 한 번의 보상을 획득할 수 있다. 이는 에이전트가 목표를 달성하기 위한 행동에 대하여 즉각적이고 명시적인 보상을 달성하는 것과 다른 희소 보상이다.

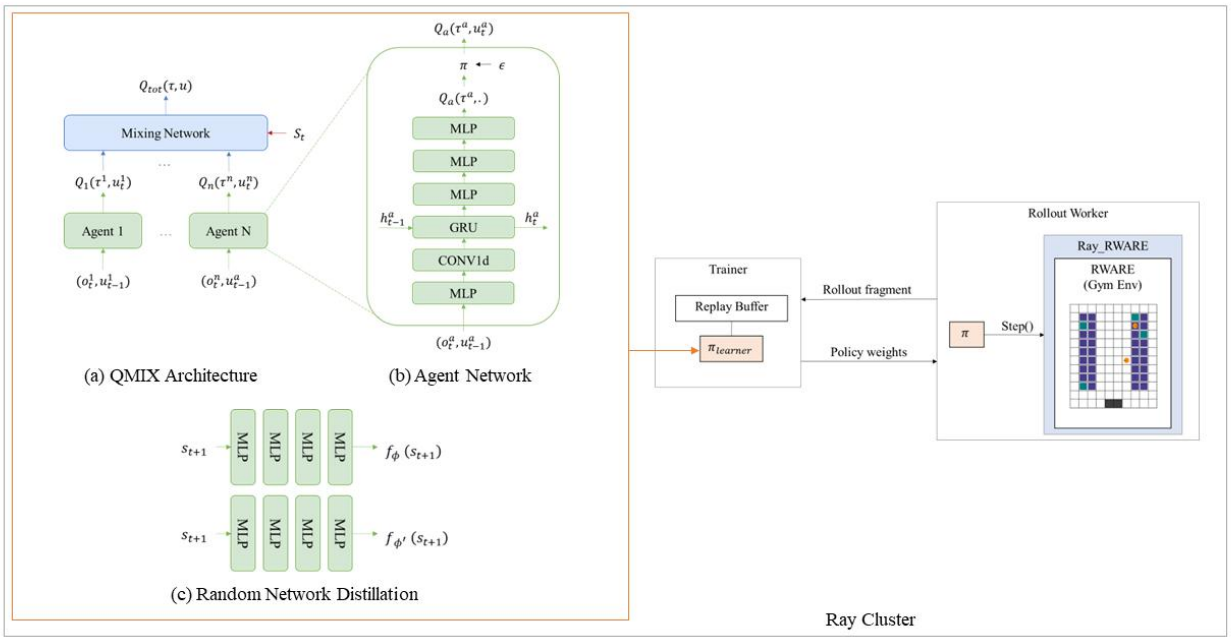

Fig. 2. Learning architecture using Ray RLlib and RWARE

2. Learning Architecture

Ray RLlib 라이브러리를 사용하여 학습하기 위해 RWARE 시뮬레이터 변경을 제안한다[그림2]. RWARE가 OpenAI Gym의 구조로 구성되어 Ray의 Standard Environment 구조로 진행했으며, 제공하는 QMIX 알고리즘으로 학습하기 위해 MultiAgentEnv 클래스를 상속받아 Ray_RWARE 클래스를 구성한다. QMIX 사용을 위해 action_space_sample과 with_agent_groups 함수를 포함한 총 7개의 함수[표2]를 재정의 및 구현한다.

Table 2. Class Ray_RWARE function

Function	OpenAI Gym Dependency	Ray RLlib Dependency
reset	O	O
step	O	O
render	O	O
close	O	O
seed	O	O
action_space _sample	X	O
with_agent _group	X	O

reset 함수는 RWARE 에피소드를 재시작하여, 각 에이전트의 관측값을 반환한다. step 함수는 각 에이전트의 행동을 환경에 적용하여, 다음 관측값과 보상값을 반환받는다. render 함수와 close 함수는 각각 환경의 시각화와 종료를 진행하고, seed 함수는 환경의 랜덤성을 추가한다. action_space_sample 함수는 에이전트 모델을 통해 확률적인 행동을 반환하고, with_agent_groups 함수는 모든 에이전트의 보상이 하나의 그룹의 보상으로 인식하게 한다.

Ray RLlib 라이브러리에서 지원하지 않는 RND는 Exploration 클래스를 상속받은 RND 클래스를 구성한다[그림3]. 크게 4가지 함수로 구성하며, 주요 함수는 Ray RLlib의 Exploration 클래스의 함수를 오버라이딩하여 구성하며, RND의 코드를 참고한다[12].

```
# Ray/rllib/utils/exploration/curiosit_rnd.py

Class RND(Exploration):
  def __init__():
        # set configuration variable
        # create RND random network
        # create RND prediction network
  def get_exploration_action():
        # override Exploration
        # delegate to sub-exploration modeul
  def get_exploration_optimizer():
        # override Exploration
        # put optimizer for RND prediction network
  def postprocess_trajectory():
        # override Exploration
        # calculate loss and update RND using optimizer
        # calculate intrinsic reward, total reward
```

Fig. 3. Class RND in Ray RLlib

Rollout Worker에서 생성된 데이터(Rollout Fragment)가 Trainer의 Replay Buffer로 전달되며, Trainer의 정책(π_{learner})은 Replay Buffer에서 배치 사이

즈만큼씩 학습한다. Trainer의 정책은 주기적으로 Rollout Worker의 정책(π)을 업데이트한다. 두 정책 구조에는 QMIX의 모델과 RND의 모델이 모두 포함된다.

3. Model Architecture

학습 구조에 포함되어 있는 신경망 모델 파라미터는 QMIX와 관련된 θ,θ^-와 RND와 관련된 $\phi,\phi^{'}$로 총 $\theta,\theta^-,\phi,\phi^{'}$ 로 구성되며, QMIX와 RND의 모델은 서로 다르게 구성한다[그림 2]. QMIX는 크게 중앙 행동-가치함수인 Q_{tot}를 학습하는 Mixing Network와 에이전트별 행동-가치 함수인 Q_a를 학습하는 Agent Network로 구성된다[그림2 (a)]. 중앙 행동-가치 함수 Q_{tot}의 argmax는 각 에이전트 행동-가치 함수 Q_a에서 argmax를 수행하는 결과의 합과 같으며, 이는 각 에이전트 a가 분산 실행될 때, 각 Q_a를 만족하는 행동을 선택하는 것을 가능하게 한다.

Q_{tot}는 Q_a의 출력값을 입력값으로 가지며, Q_a의 단조 증가 함수라는 조건에 따라, 가중치를 0 또는 양의 값을 가져야 한다. Q_{tot}의 각 계층은 입력으로 하이퍼 네트워크(Hyper Network)의 출력값을 받는데, 이때 하이퍼 네트워크의 입력값으로 전체 상태(Global State)를 받는다. 이 하이퍼 네트워크는 Mixing Network의 가중치가 음이 아닌 값을 갖는 것을 보장한다.

QMIX의 손실함수는 수식 (1)를 따르며, b는 Replay Buffer에서의 배치 사이즈, θ^-는 타겟 네트워크(Target Network), 그리고 $y^{tot}=r+\gamma\max_{u'}Q_{tot}(\tau',u',s';\theta^-)$를 의미한다.

$$L(\theta)=\sum_{i=1}^{b}\left[\left(y_i^{tot}-Q_{tot}(\tau,u,s;\theta)\right)^2\right] \quad (1)$$

중앙 행동-가치함수인 Q_{tot}는 기존 QMIX와 동일한 구조를 사용하며, Q_a에 대한 모델을 수정하여 진행한다[그림2(b)]. 기존 QMIX Agent Network는 완전 연결 계층(Fully Connect Layer)사이에 GRU를 추가한 모델 구조이다. 제안 구조는 GRU 전에 Conv1D 계층을 추가하여, 데이터에 대한 시간 종속성을 반영[13]한 특징을 추출할 수 있게 구성했으며, GRU 이후 완전 연결 계층을 추가했다. Conv1D 계층을 추가함으로써 추후 실행 시, Q_a모델이 부분 관측값에 대하여 지역적인 패턴 추출을 감지하고, 시간에 따라 순차적으로 획득하는 에이전트들의 부분 상태 값에 대하여 시간적인 특징을 추출하여 학습하고자 했다. 또한 FC 계층을 추가하여 Conv1D 계층과 GRU 계층에서 추출한 부분 특징이 반영될 수 있게 했다.

에이전트의 내적 보상은 두 개의 동일한 신경망을 사용하여[그림2(c)], s_{t+1}에 대한 입력값으로 특징을 추출하게 한다. 예측 신경망(ϕ)이 수식 (2)를 통해 업데이트되며, 평균제곱오차 형태의 손실함수를 최소화하도록 학습된다. 무작위 신경망($\phi^{'}$)은 학습 초기에 무작위로 얻은 임의의 값으로 고정된다.

$$L(\phi)=\sum_{t=1}\frac{1}{2}\left(f_{\phi}(s_{t+1})-f_{\phi^{'}}(s_{t+1})\right)^2 \quad (2)$$

내적 보상 r_{t+1}^i은 수식 (3)처럼 정의되며, 내적 보상과 함께 에이전트가 얻는 보상 r_{t+1}은 수식 (4)처럼 실제 환경으로부터 획득한 외적 보상 r_{t+1}^e와 내적 보상r_{t+1}^i으로 정의한다.

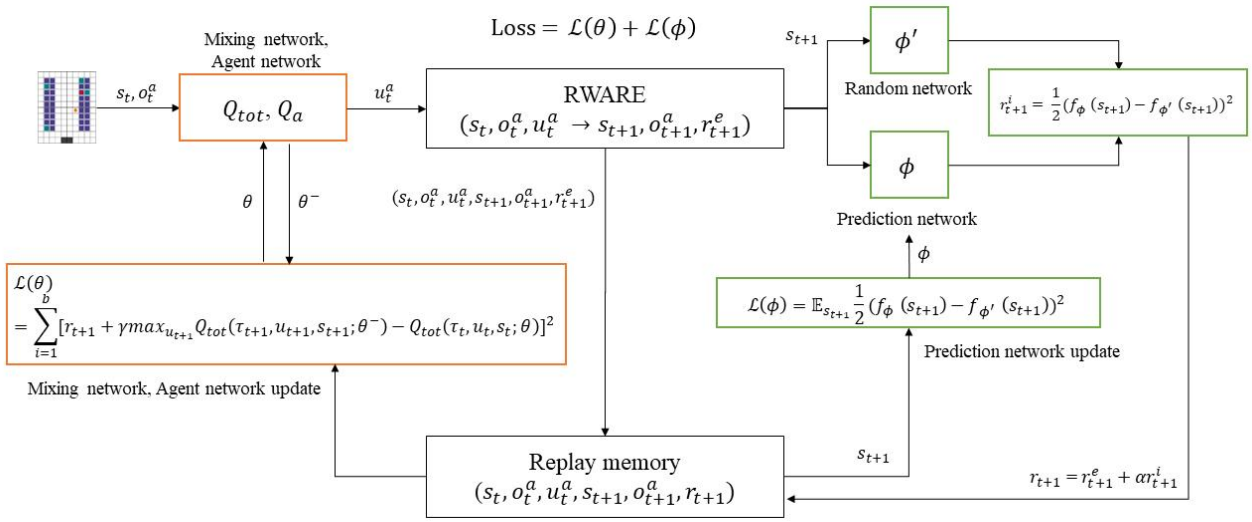

Fig. 4. Structure of the entire learning process(QMIX + RND)

$$r_{t+1}^{i} = \frac{1}{2}\left(f_{\phi}(s_{t+1}) - f_{\phi'}(s_{t+1})\right)^2 \quad (3)$$

$$r_{t+1} = r_{t+1}^{e} + \alpha r_{t+1}^{i} \quad (4)$$

전체 모든 신경망의 학습 과정 구조는 그림4과 같으며, 전체 손실함수는 QMIX와 RND를 고려하여 수식 (5)로 정의한다.

$$Loss = L(\theta) + L(\phi) \quad (5)$$

IV. Experiments

1. Experiment Environment

Ray RLlib의 QMIX 알고리즘을 사용하여, 실험 환경을 구성한다. 에이전트의 부분 관측값은 71 1D 벡터로 구성되며, 자신의 정보 8개, 관측 범위 내 에이전트 정보 45개, 관측 범위 내 선반 정보 18개로 구성된다. 이때 각 에이전트는 위치한 셀을 포함한 9개의 관측 정보를 획득할 수 있으므로, 관측 범위 내 에이전트 정보는 관측 그리드 셀(9) × 에이전트 정보(5), 관측 범위 내 선반 정보는 관측 그리드 셀(9) × 선반 정보(2)로 나타낸다.

자신의 정보에는 x, y 좌표 정보(2), 선반을 들고 있는 상태 유무 정보(1), 방향 정보(4)를 포함하며, 관측 범위 내 에이전트 정보는 에이전트 존재 유무 정보(1), 에이전트 방향 정보(2)를 포함한다. 또한 선반 정보에는 선반 유무 정보(1), 요청된 선반 유무 정보(1)가 표시된다.

Mixing Network의 전체 상태는 에이전트 두 개의 관측 정보를 길이가 32인 1D 벡터로 임베딩한 상태값을 사용한다. 해당 Mixing Network는 학습 시에만 사용하며, 추론 시 사용하지 않는다. 기존 QMIX와 동일하게 모든 GRU의 hidden state은 64로 구성하며, 모델의 입력 계층과 출력 계층을 제외한 모든 계층은 input dimension과 output dimension을 64로 동일하게 구성한다. Conv1D kernel size 3으로 사용하며, padding을 1 추가한다.

RND의 신경망은 input dimension으로 에이전트들의 관측값을 142 1D 벡터로 구성하며, output dimension은 1로 hidden dimension은 64로 구성한다. RND의 mask update rate를 0.5로 설정하여 loss 계산을 진행했으며, 외적 보상과 더해지는 과정의 상수는 1로 고정했다.

학습 시 사용한 파라미터[표 3]는 모두 동일하게 진행했으며, 2.5M 스텝동안 학습을 진행했다. 추론 시, 동일하게 모두 RWARE max steps를 3K, RWARE inactivity steps를 0으로 설정하여, 총 150K 번의 스텝으로 실험 평가를 진행했다.

Table 3. Hyper parameters for training

Learning rate	0.003
Train batch size	64
Exploration final epsilon	0.01
Exploration epsilon anneal	1000000
Target network update frequency	100
Max sequence length	64
Mixing embedding dimension	32
Replay buffer capacity	10000
Agent observation size	71
Gamma	0.99
RWARE max steps	6000
RWARE inactivity steps	1000
RND constant a	1
RND mask update rate	0.5

2. Experiment Evaluation Criteria

에이전트가 지속해서 희소 보상을 얻기 위해서는 3가지의 상태를 순차적으로 전이해야 한다. 이에 따라 각 에이전트가 순차적으로 3가지 상태에 도달하는 지를 실험 결과를 통해 분석한다. 실험 평가 상태 항목은 아래와 같다.

(1) 에이전트가 요청한 선반을 드는 상태
(2) 에이전트가 요청한 선반을 들고 목적지로 도착하는 상태
(3) 에이전트가 요청한 선반을 들고 목적지에 도착한 후, 다시 비어있는 자리에 선반을 내려놓는 상태

또한, 에이전트 간 협력의 지표로 에이전트 간 충돌 횟수를 측정하여 평가한다.

3. Experiment Results

학습 결과[그림5]는 제안한 QMIX(proposal)+RND 학습 구조가 에이전트의 탐험(Exploration)으로 더 많은 희소 보상을 얻으며 학습한 것을 확인할 수 있다. RND를 사용하지 않는 QMIX(proposal)과 QMIX(default)는 Exploration epsilon anneal에 따라 수렴을 위해 1M 스텝 후로 탐험 확률이 감소하게 되어, 이후로 에피소드 외적 보상 최대치(Episode extrinsic reward max)와 에피소드 외적 보상 평균(Episode extrinsic reward mean)이 모두 0으로 감소하며 학습한다. 이에 반해 QMIX(proposal) + RND은 에피소드 외적 보상 최대치가 대부분 경우 1을 얻으며, 최대 희소 보상을 한 에피소드에서 2를 획득하며 학습하는 것을 볼 수 있다. 또한, 에피소드 외적 보상 평균에서도 2.5M 스텝까지 지속해서 보상을 획득하며 학습하는 것을 그래프에서 확인할 수 있다.

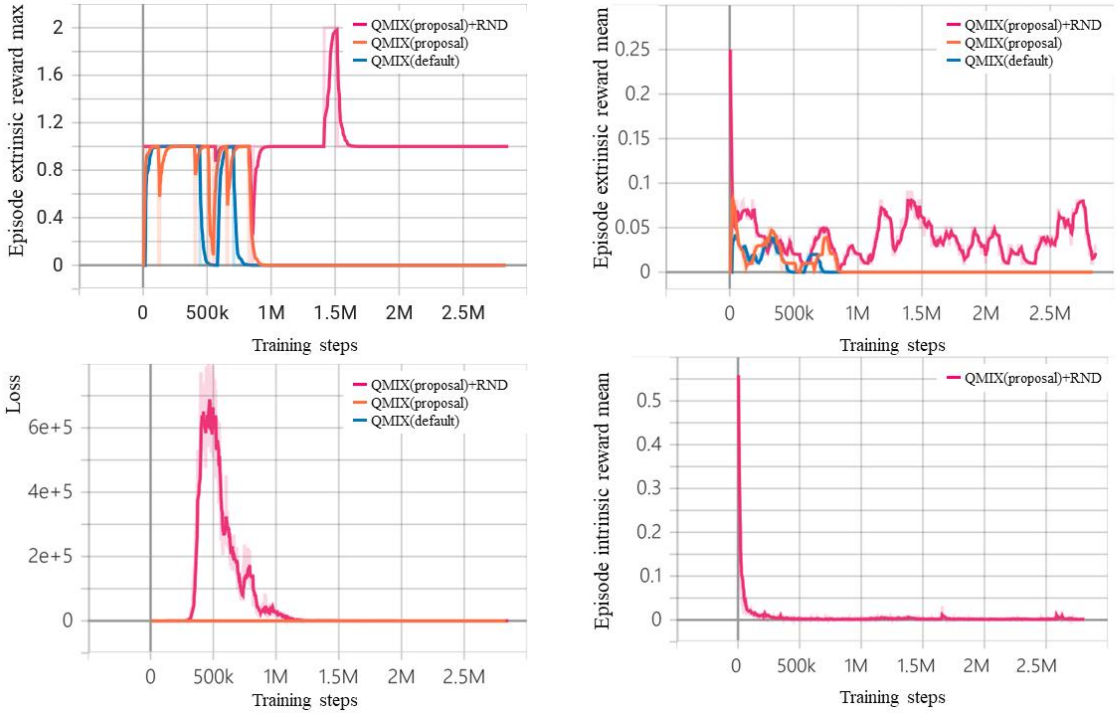

Fig. 5. Training result graphs (2.5M steps)

Table 4. Inference result (150K steps)

150K steps	Load requested shelf	Arrive destination	Unload requested shelf	Collisions
QMIX(default)	213	13	9	**603**
QMIX(proposal)	**256**	15	9	696
QMIX(proposal) + RND	234	**18**	**10**	669

QMIX(proposal)+RND는 1M 스텝 전까지 손실(Loss)이 크게 발생하지만, QMIX(proposal)과 QMIX(default)와 마찬가지로 추후 0으로 수렴하는 것을 손실(Loss) 그래프에서 확인 할 수 있다. 에피소드 내적 보상 평균(Episode intrinsic reward mean)은 QMIX(proposal)+RND에만 해당하는 그래프로, 에이전트가 학습 시, RND를 통해 얻는 내적 보상을 나타낸다. 새로운 경험을 많이 하게 되는 500K 스텝 전에 내적 보상이 높은 것을 볼 수 있으나, 차츰 2.5M 스텝까지 감소하며 수렴함을 확인할 수 있다.

표4는 150K 스텝의 추론 결과를 실험 평가 기준에 따라 에이전트 상태 변이와 에이전트 간 충돌 횟수를 실험한 결과다. QMIX(proposal)은 QMIX(default)보다 요청된 선반을 드는 경우(Load requested shelf)와 목적지에 도착(Arrive destination)하는 경우가 증가하여, QMIX의 Agent Network에 추가한 Conv 계층과 FC 계층이 에이전트의 부분 관측값의 시간적 특징을 추출 및 반영하여 순차적으로 상태 전이 부분이 학습된 결과로 볼 수 있다. 하지만, 서로 간 충돌(Collisions)이 증가하였다. 제안한 학습 구조인 QMIX(proposal)+RND는 요청한 선반을 목적지에 가장 많이 배달하는 것을 확인할 수 있으며, 배달한 선반을 다시 빈공간을 찾아서 선반을 내려놓는 상태(Unload requested shelf)도 다소 증가함을 확인할 수 있다. 또, 요청 선반을 드는 행위 대비 요청 선반을 많이 배달하여, 효과적으로 동작할 가능성을 확인했다. 또한 QMIX(proposal)에 비해 희소 보상 대비 충돌 횟수가 다소 감소함을 확인했으며, 이는 다른 에이전트와의 협력을 고려한 결과라고 볼 수 있다.

이러한 RWARE 환경에서의 실험 결과를 통해 유사한 전장 교전 상황에서의 아군 에이전트 간 협업 진행이 가능할 것으로 예상되며, 아군 전투 승리 또는 아군 무장 발사에 따른 적 객체 격추와 같은 희소 보상에 대해서도 QMIX(proposal)+RND 학습 구조를 통해 학습 및 추론 가능할 것으로 예측된다.

V. Conclusions

본 논문에서는 희소 보상이 획득되는 전장 교전 상황에서 아군 에이전트 간 협업 진행 가능성을 확인하기 위해, 희소 보상 환경인 Multi-Robot Warehouse Environment(RWARE) 시뮬레이터를 사용하여 전장 교전 도메인과 유사한 상황을 정의한다. 정의한 문제 상황에서 에이전트 간 협업을 위해 QMIX 알고리즘을 사용했으며, Agent Network를 개선하여 부분 관측값에 대한 패턴과 시간 특징을 추출하게 한다. 또한, 에이전트의 희소 보상에 대한 탐험 증가를 위해 RND를 적용하여, 전장 교전 도메인 지식에 독립적으로 희소 보상 획득 개선이 가능함을 확인한다. 이를 강화학습 Ray RLlib를 사용하여 QMIX와 RND 학습 구조를 구성했으며, 해당 구조에서 정의한 RWARE의 희소 보상 문제를 정의한 평가 기준으로 학습하여 실험 진행했다. 실험을 통해 Agent Network가 에이전트 부분 관측값의 시간적 특징이 추출되어 학습 진행된 것을 확인하였으며, RND가 희소 보상 획득에 도움이 되고 이로 인한 추론 성능의 개선도 관찰되었다. 향후 연구는 본 연구의 가능성을 바탕으로, 실제 전장 교전 상황과 유사한 시뮬레이터에서 희소 보상 협업 연구를 진행한다.

REFERENCES

[1] C. G. Lee, J. U. Baek, J. N. Son, S. Y. Lee and Y. G. Ha, "Multi-agent based Manned/unmanned Collaboration System for Combatant's Battlefield Situation Awareness," Journal of the Institute of Electronics and Information Engineers, Vol. 59, No. 2, pp. 126-134, Feb. 2022. DOI: 10.5573/ieie.2022.59.2.126

[2] U. H. Choi and J. M Ahn, "A Study on Multi-agent Deep Reinforcement Learning for Distributed Cooperative Multi-UAV Mission," in Proc. of the KSAS 2021 Fall Conf. , pp. 958-959, Jeju Island, Korea, Nov 2021.

[3] A. P. Pope, J. S. Ide, D. Micovic, H. Diaz, D. Rosenbluth, L. Ritholtz, J. C. Twedt, T. T. Walker, K. Alcedo, D. Javorsek, "Hierarchical Reinforcement Learning for Air-to-Air Combat," In Proc. of the International Confernse on Unmanned Aircraft System, June 2021. DOI: 10.48550/arXiv.2105.00990

[4] M. H. Park, J. H. Oh, C. Y. Kim, H. J. Seol, "The Specification of Air-to-Air Combat Tactics Using UML Sequence Diagram," Journal of the KIMST, Vol. 24, No. 6, pp. 664-675, 2021. DOI: 10.9766/KIMST.2021.24.6.664

[5] G. Papoudakis, F. Christianos, L. Schäfer and S. V. Albrecht, "Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks," in Proc. of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), 2021. DOI: 10.48550/arXiv.2006.07869

[6] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX : Monotonic Value Funtion Factorisation for Deep Multi-Agent Reinforcement Learning," in Proc. of the 35th International Conference on Machine Learning, Stockholm, Sweden, 2018. DOI: 10.48550/arXiv.1803.11485

[7] Y. Burda, H. Edewards, A. Storkey, O. Klimov, "Exploration by Random Network Distillation," 2018. DOI: 10.48550/arXiv.1810.12894

[8] Ray Project, Anyscale, *https://github.com/ray-project/ray*

[9] M. Samvelyan, T. Rashid, C. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C. M. Hung, P. H. S. Torr, J. Foerster and S. Whiteson, "The StarCraft Multi-Agent Challenge," in Proc. Deep Reinforcement Learning at the 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2019. DOI: 10.48550/arXiv.1902.04043

[10] Tan, M, "Multi-Agent Reinforcement Learning Independent vs Cooperative Agents" In Proc. of the Tenth International Conference on Machine Learning, pp. 330-337, 1993.

[11] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K.l Tuyls and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward," In Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems, 2017. DOI: 10.48550/arXiv.1706.05296

[12] Random network distillation, OpenAI, *https://github.com/openai/random-network-distillation*

[13] H. J. Kim, K. S. Kim, S. Y. Hwang and J. H. Lee, "The Fault Diagnosis Model of Ship Fuel System Equipment Reflecting Time Dependency in Conv1D Algorithm Based on the Convolution Network," J. Navig. Port Res, vol. 46, no. 4, pp. 367-374, August 2022. DOI: 10.5394/KINPR.2022.46.4.367

Authors

Minkyoung Kim received the B.S. degree in Computer Science and Engineering from Hanyang University, Korea, in 2015 and the M.S. degree in Computer Science and Engineering from Pohang University of Science and Technology, Korea, in 2017. Minkyoung Kim is currently an engineer in Hanwha Systems. She is interested in Deep Learning, Deep Reinforcement Learning and Security.