QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

2023. 04. 24

김정인

QMIX (2018, ICML)

❖ QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

- 2018년 ICML에 게재된 논문으로, 23.04.23 기준 1344회 인용됨
- Monotonicity 제약 조건과 추가적인 상태 정보를 사용해 2018년 이전에 MARL 방법론 중 가장 우수한 성능을 보여줌
- StarCraft 2 학습 환경을 사용하는 대표적인 Centralised Training Decentralised Execution (CTDE) 방법론

QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

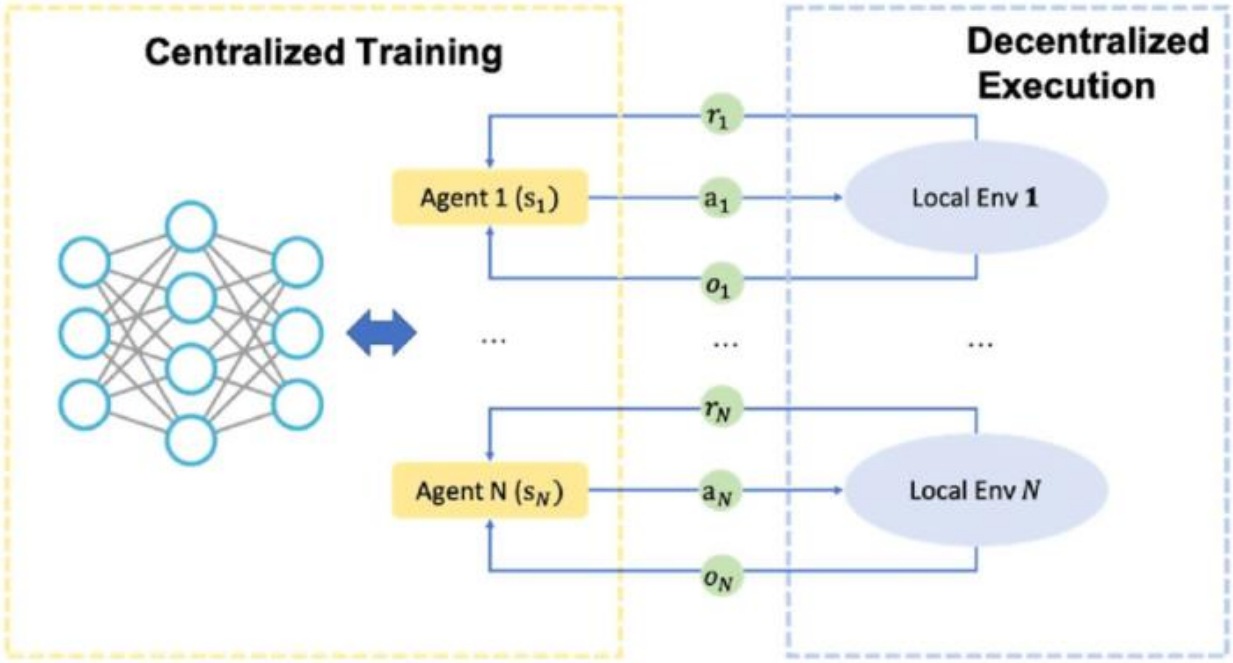
---

Tabish Rashid [*1] **Mikayel Samvelyan** [*2] **Christian Schroeder de Witt** [1]
Gregory Farquhar [1] **Jakob Foerster** [1] **Shimon Whiteson** [1]

Research Purpose

❖ QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

- 여러 에이전트를 통해 나오는 공동 행동 가치 함수(보상)을 최대화하기 위한 centralized training하고 decentralized policy로부터 추출하는 최고의 행동에 대한 방법은 불분명한 것에서부터 시작됨
  - ✓ CTDE는 여러 에이전트간 의 협력의 어려움, 환경의 불안정성으로 인해 공동의 보상을 최대화하기 위한 여러 에이전트의 최적 행동을 학습하기 위해 연구됨



Proposed Methodology

❖ QMIX: Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning

1. 단순하게 Factorization하는 것이 아닌, 각 에이전트가 최대 Q값을 갖도록 하는 행동을 취했을 때의 결과가 결국 전체 누적 보상을 최대화하는 Factorization 시행
2. 전체 Q값과 각 에이전트의 Q값 마다 monotonic 관계가 성립하도록 하는 제약 조건을 추가
   - ✓ 이를 통해, 각 에이전트가 공동 행동 가치 함수(공동 보상)를 최대화하기 위한 효과적인 행동(협력) 학습 가능

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}$$

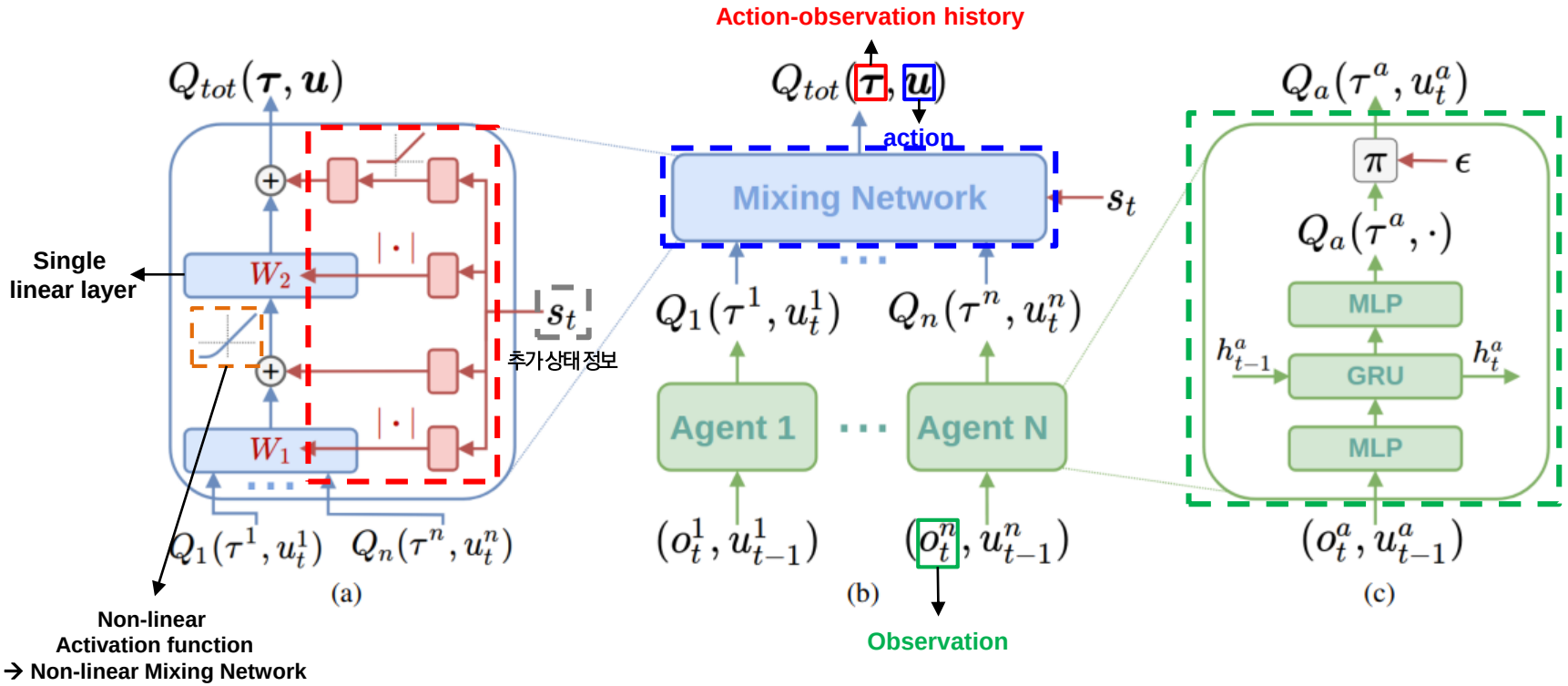
$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A.$$

Proposed Methodology

❖ QMIX: Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning

- Q_{tot}을 나타내기 위해 agent networks, mixing network, hypernetwork 사용됨

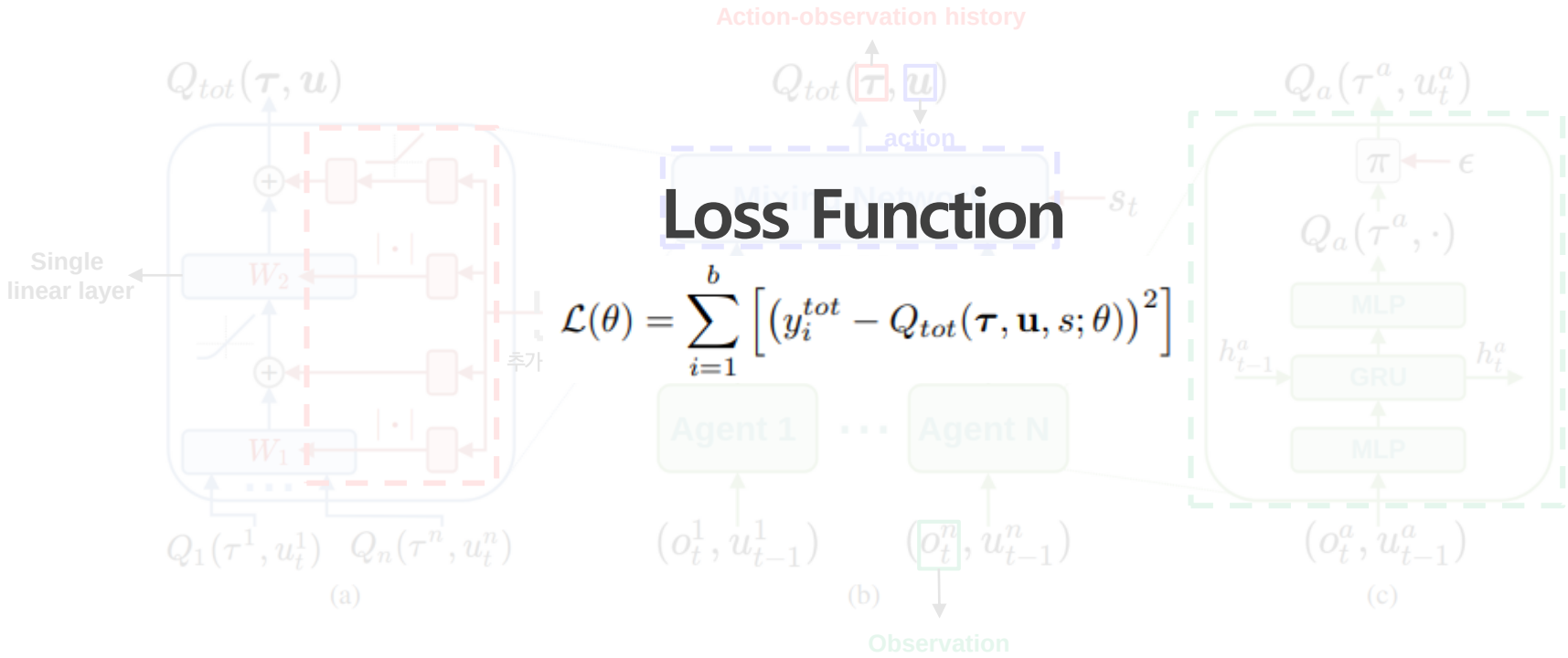
1. Agent network: Deep Recurrent Q-Learning Network (DRQN) 사용 / 현재 관측 값과 마지막 행동이 입력으로 사용됨
2. Mixing network: feed-forward neural network 사용 / agent network의 출력 값을 입력으로 사용 → mixing **monotonically**
3. Hypernetwork: mixing network의 파라미터를 생성해주는 네트워크 / 추가 상태 정보를 입력으로 사용



Proposed Methodology

❖ QMIX: Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning

- Q_{tot}을 나타내기 위해 agent networks, mixing network, hypernetwork 사용됨
  1. Agent network: Deep Recurrent Q-Learning Network (DRQN) 사용 / 현재 관측 값과 마지막 행동이 입력으로 사용됨
  2. Mixing network: feed-forward neural network 사용 / agent network의 출력 값을 입력으로 사용 → mixing **monotonically**
  3. Hypernetwork: mixing network의 파라미터를 생성해주는 네트워크 / 추가 상태 정보를 입력으로 사용



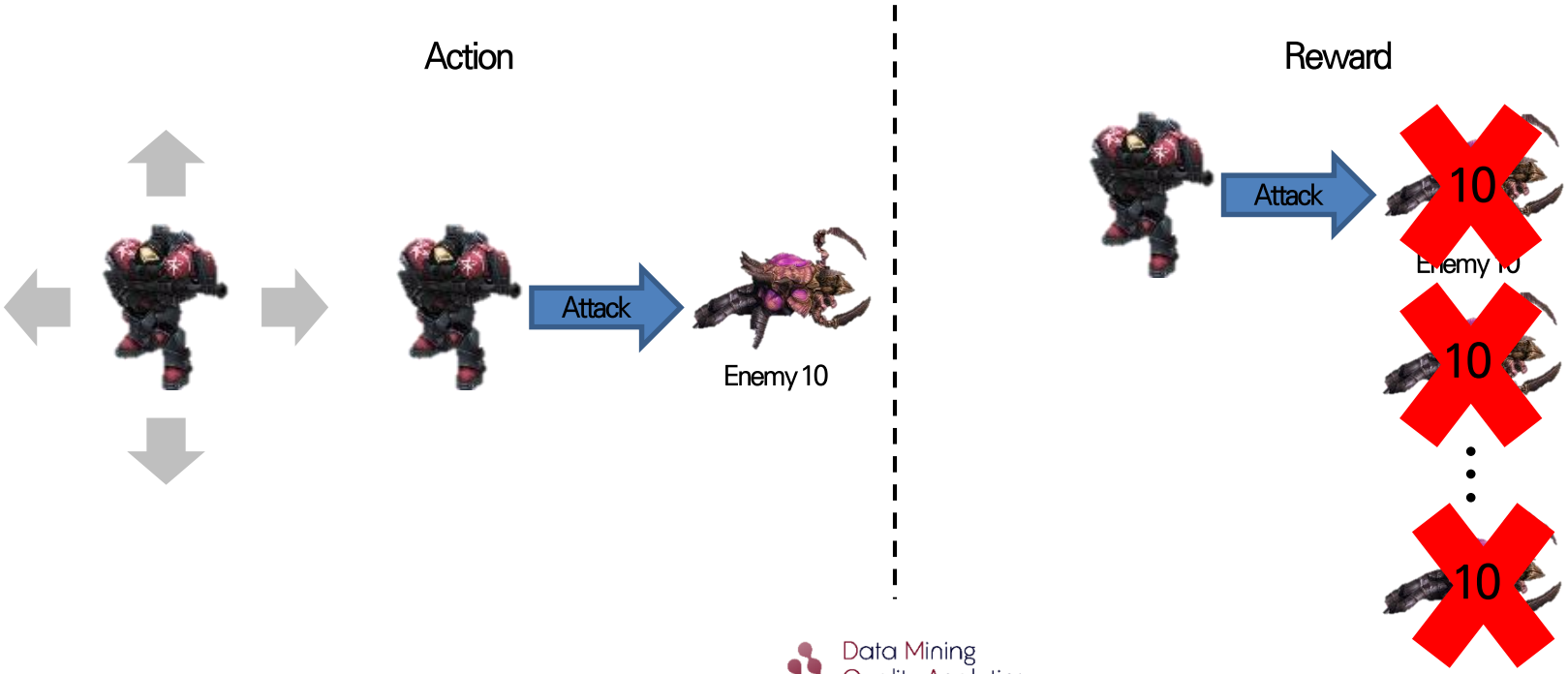
Loss Function

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

Experiment

❖ Main experiment

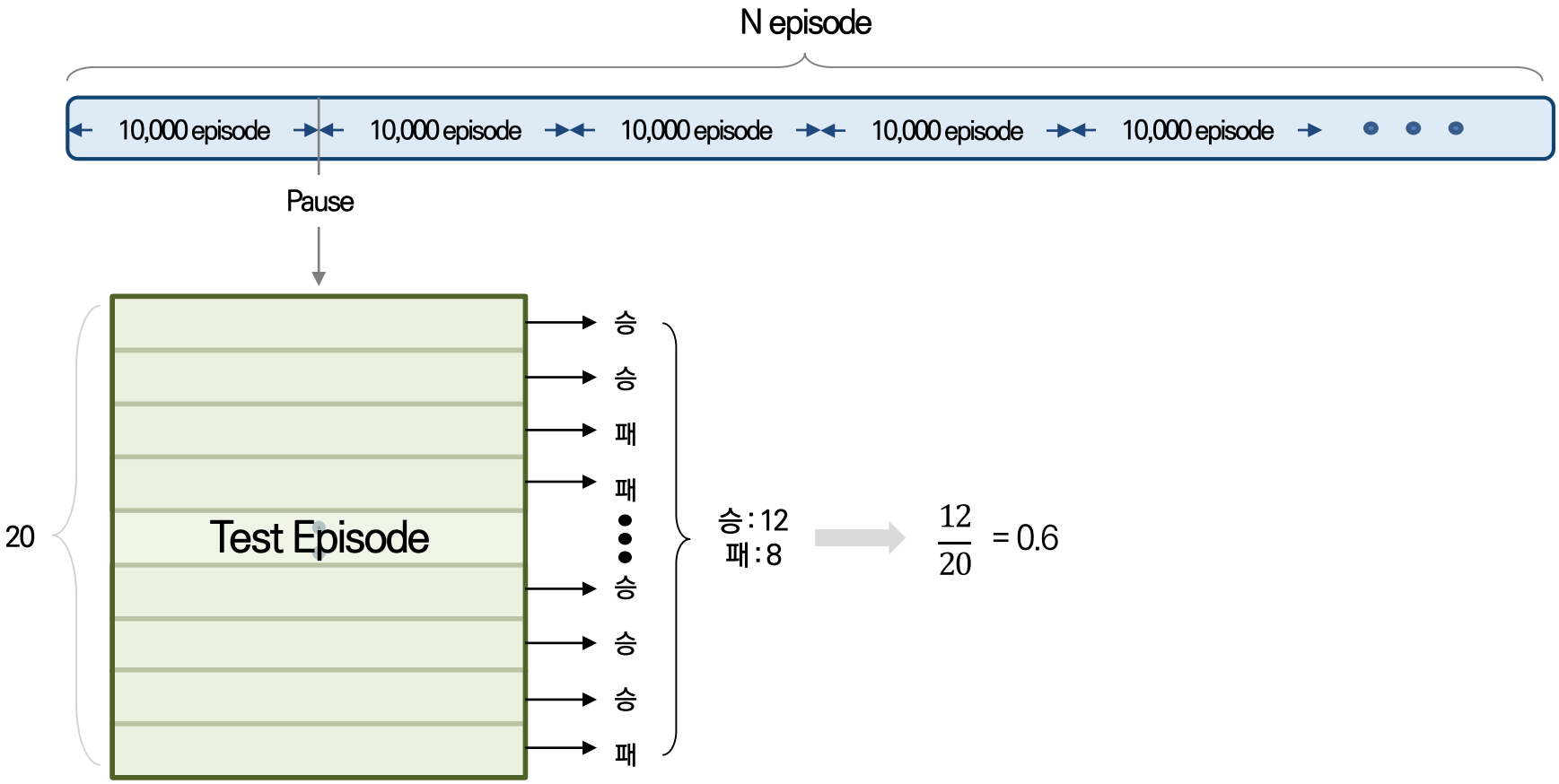
- 아군과 적군 모두 동일한 수의 유닛 사용: 3m, 5m, 8m, 2s_3z, 3s_5z, 1c_3s_5z (총 6개 시나리오)
  - ✓ 에피소드마다 유닛의 초기 위치는 다양함
  - ✓ 에이전트의 움직임: move[north, south, east, west), attack[enemy_id], stop, noop(아무런 행동 정보 없음)
- 각 time step마다 적군 유닛에게 피해를 입힌 만큼 보상을 얻음
  - ✓ 각 적군을 죽이면 10점 / 모든 적군을 섬멸하면 200점 → 최대 누적 보상(upper bound)을 보장하기 위해 20점으로 정규화



Experiment

❖ Main experiment

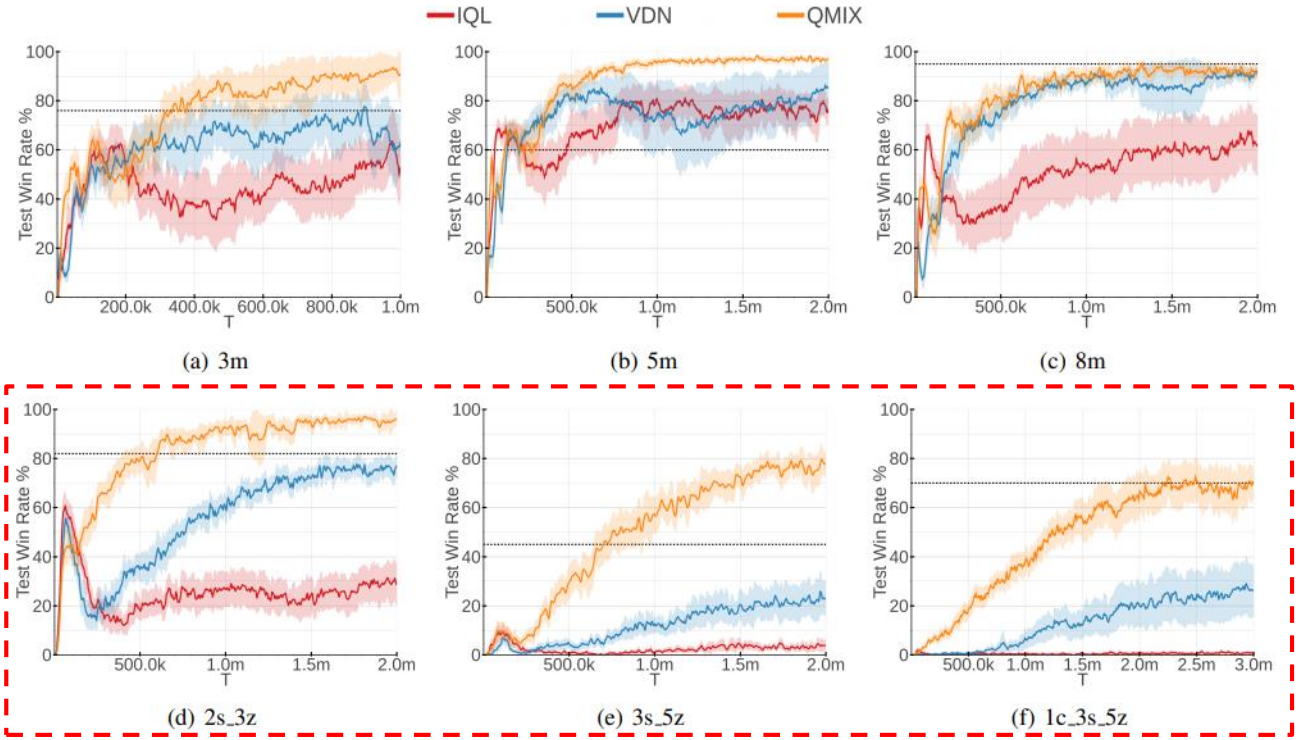
- 평가 방법: 학습하는 도중에 100개의 episode마다 중지하고 20개의 독립적인 에피소드에 대한 승률 계산



Experiment

❖ Main experiment

- 제안 방법론(QMIX)가 가장 우수한 성능을 보임
- IQL 방법의 경우 공동 보상이 아닌 독립적인 에이전트의 보상을 활용해 각 에이전트 학습 진행
  - → VDN과 QMIX와 같은 공동 보상을 활용하는 방법론이 우수한 성능을 보임 ➔ 공동 보상 활용의 장점 입증
- 유닛의 종이 여러 개인 시나리오에서도 QMIX가 더 우수한 성능을 보임 → 추가 상태 정보를 활용 덕분

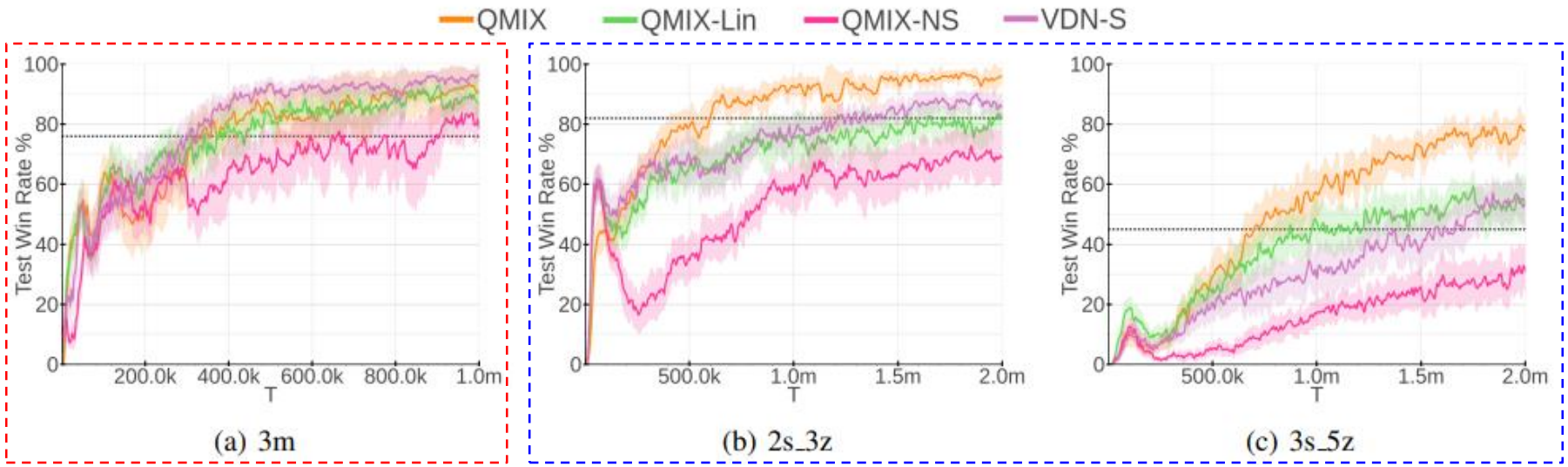


Experiment

❖ Ablation Study

- 추가 상태 정보 활용과 mixing network에서 비선형성 부여의 효과에 관한 추가 실험

1. QMIX_NS: 추가 상태 정보를 활용하지 않은 방법론
2. QMIX_Lin: mixing network에서 비선형성을 제거한 방법론
3. VDN_S: 추가 상태 정보를 활용하고 linear-factorization을 사용하는 방법론



- 1. 동일한 유닛을 사용했을 때, non-linear factorization을 사용하는 것이 꼭 우수한 성능을 보이는 것은 아님을 보여줌
  → (3) VDN_S vs other networks

- 서로 다른 유닛을 사용했을 때, QMIX(추가 상태 정보 및 non-linear factorization을 활용)가 가장 우수한 성능을 보임
- 추가 상태 정보를 활용하지 않고 non-linear factorization만을 사용했을 때 항상 우수한 것은 아님 (VDN vs QMIX_NS)
  → QMIX_NS가 VDN보다 아주 조금 더 좋은 성능을 보임
- 추가 정보를 완전히 활용했을 때 non-linear factorization을 사용해야하는 필요성을 보여줌 (QMIX_Lin vs VDN_S)
  → (b)와 (c)에서 두 방법론의 성능이 오락가락 함