Resource Competition in Multi-Agent Reinforcement Learning:

A Literature Review

Anastasia Chernavskaia, Moritz Peist, Nicolas Rauth

June 29, 2025

Barcelona School of Economics

Outline

Introduction to Multi-Agent Reinforcement Learning

What is Multi-Agent Reinforcement Learning (MARL)?

- Extension of single-agent RL to environments with **multiple learning agents**
- Agents must learn to interact, coordinate, or compete with each other
- Each agent's optimal policy depends on other agents' policies
- Introduces challenges: non-stationarity, coordination, credit assignment

Key Characteristics:

- Simultaneous learning and adaptation
- Emergent behaviors from agent interactions
- Applications: robotics, autonomous vehicles, game theory, economics

Adversarial Environments

Simple Adversary Environment

Environment Setup:

- **Agents:** 1 adversary + 1 cooperative agent
- **Landmarks:** Fixed black landmarks in the environment
- **Objective:** Adversary tries to reach landmarks, agent tries to prevent it

Competitive Dynamics:

- Zero-sum game structure
- Adversary reward $= -$ Agent reward
- Tests robustness of learned policies
- Requires strategic thinking and adaptation

PPO Algorithm in MARL

Proximal Policy Optimization (PPO)

Why PPO for MARL?

- **Stability:** Prevents large policy updates that could destabilize training
- **Sample Efficiency:** Reuses experience through multiple epochs
- **Simplicity:** Easy to implement and tune compared to TRPO

PPO Objective Function:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ and $\epsilon = 0.2$

PPO Hyperparameters in Our Implementation

Network Architecture:

- **Hidden layers:** [256, 256, 128] neurons
- **Activation:** Hyperbolic tangent (Tanh)
- **Policy:** Actor-Critic with shared features

Training Parameters:

- **Learning rate:** 3×10^{-4}
- **Batch size:** 512, **Steps per update:** 256
- **Discount factor:** $\gamma = 0.98$
- **GAE lambda:** $\lambda = 0.95$
- **Entropy coefficient:** 0.01 (encourages exploration)

Environment Preprocessing

PettingZoo and SuperSuit Wrappers

Environment Pipeline:

1. **PettingZoo MPE:** Multi-Particle Environment framework
2. **Black Death:** Removes dead agents from observation/action spaces
3. **Pad Observations:** Ensures consistent observation dimensions
4. **Flatten:** Converts structured observations to flat vectors
5. **AEC to Parallel:** Transforms turn-based to simultaneous actions
6. **Vectorization:** Enables batch processing for efficiency

Benefits:

- Standardized interface for RL algorithms
- Improved computational efficiency

Training and Evaluation

Training Process

Training Configuration:

- **Total timesteps:** 500,000
- **Environment:** Continuous action space
- **Monitoring:** Weights & Biases (wandb) integration
- **Logging:** TensorBoard for real-time metrics

Key Metrics Tracked:

- Episode rewards (individual and collective)
- Policy and value function losses
- Episode lengths and success rates
- Agent coordination measures

Visualization and Analysis

Evaluation Methods:

- **Animated GIFs:** Show agent behavior evolution
- **Training dashboards:** Real-time metric visualization
- **Performance plots:** Reward trends and learning curves

Behavioral Analysis:

- Agent positioning strategies
- Adversarial adaptation patterns
- Coordination emergence over time
- Robustness to opponent strategies

Implementation Highlights

Technical Implementation

Key Libraries and Frameworks:

- **Stable-Baselines3:** PPO implementation
- **PettingZoo:** Multi-agent environment suite
- **SuperSuit:** Environment preprocessing wrappers
- **PyTorch:** Neural network backend
- **Weights & Biases:** Experiment tracking

Code Structure:

- Modular design for reusability
- Custom visualization library
- Reproducible experimental setup
- Comprehensive logging and monitoring

Challenges and Solutions

MARL Challenges Addressed

Non-stationarity:

- **Problem:** Environment changes as other agents learn
- **Solution:** PPO's stable policy updates + experience replay

Credit Assignment:

- **Problem:** Which agent caused the reward?
- **Solution:** Individual rewards + shared value function

Scalability:

- **Problem:** Exponential growth in joint action space
- **Solution:** Vectorized environments + parallel training

References

References i