Reproduction and Improvement of CPPO, MAPPO, and IPPO Algorithms in Transport Task

VMAS Multi-Agent Reinforcement Learning Experiment Report

Chen Junfan

January 2026

Contents

1 Project Code

The complete project code and implementation details are available on GitHub:
https://github.com/OrangeSeventh/RL_Assignment

2 Report Abstract

This report details the multi-agent reinforcement learning (MARL) experiments conducted on the Transport task within the VMAS (Vectorized Multi-Agent Simulator) framework. We completed the following three main tasks:

Task 1: VMAS Code Annotation

- Read the paper "VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning"
- Added detailed Chinese annotations to VMAS core code, covering 9 key files
- Annotation content highlights key concepts, data structures, and algorithm logic

Task 2: MARL Algorithm Reproduction

- Reproduced three PPO-based MARL algorithms in the Transport scenario: CPPO, MAPPO, and IPPO
- Completed comprehensive training and performance evaluation
- Verified the core conclusions of the paper

Task 3: Algorithm Improvement

- Implemented observation normalization improvement scheme based on issues found in original reproduction
- MAPPO final reward improved from -0.1255 to 0.0284 (+122.6%)
- MAPPO peak reward improved from 0.1612 to 0.6049 (+275.1%)
- Training overhead increased by only 0.2%

Experiment Dates: January 14-15, 2026
Report Version: v1.0
Author: Chen Junfan

3 VMAS Framework Introduction

3.1 VMAS Overview

VMAS (Vectorized Multi-Agent Simulator) is an open-source multi-agent reinforcement learning benchmark framework with the following core features:

- **Vectorized Physics Engine**: 2D physics engine implemented based on PyTorch, supporting large-scale parallel simulation
- **High Performance**: Compared to OpenAI MPE, VMAS can execute 30,000 parallel simulations in 10 seconds, with performance improvement exceeding 100x
- **Modular Design**: Provides 12 challenging multi-agent scenarios, supporting custom scenario development
- **Compatibility**: Compatible with mainstream frameworks such as OpenAI Gym and RLlib

3.2 Transport Task Description

The Transport task is a typical collaborative transportation scenario, requiring multiple agents to cooperate in moving one or more packages from starting positions to target positions.

3.2.1 Task Characteristics

- **Collaborative**: A single agent cannot complete the task independently, requiring multiple agents to work together
- **Physical Interaction**: Agents need to physically interact with packages (pushing)
- **Spatial Reasoning**: Agents need to understand spatial relationships and plan optimal paths
- **Dynamic Environment**: Package movement is constrained by physical laws and has inertia

3.2.2 Task Parameters

- Number of agents: 4
- Number of packages: 1
- Package mass: 50
- Package size: 0.15×0.15
- Maximum steps: 500
- Observation dimension: 11 dimensions (agent position, velocity, package relative position, package velocity, whether package is on target)
- Action dimension: 2 dimensions (force in x and y directions)

4 MARL Algorithm Principles

4.1 CPPO (Centralized PPO)

Principle: Centralized training, centralized execution

- **Training Phase**: Uses global information (observations of all agents) to train a shared policy network
- **Execution Phase**: Uses global information to generate actions
- **Advantage**: Can fully utilize global information, theoretically optimal performance
- **Disadvantage**: Requires global information during execution, high communication overhead

4.2 MAPPO (Multi-Agent PPO)

Principle: Centralized training, decentralized execution

- **Training Phase**: Uses global information to train a shared Critic network, but each agent has an independent Actor network
- **Execution Phase**: Each agent only uses local observations to generate actions
- **Advantage**: Utilizes global information during training, only needs local information during execution, balancing performance and practicality
- **Disadvantage**: Higher training complexity

4.3 IPPO (Independent PPO)

Principle: Decentralized training, decentralized execution

- **Training Phase**: Each agent independently trains its own policy network, using only local observations
- **Execution Phase**: Each agent only uses local observations to generate actions
- **Advantage**: Simple implementation, strong scalability
- **Disadvantage**: Cannot utilize global information, poor performance in collaborative tasks

5 Experimental Setup

5.1 Environment Configuration

```
ENV_CONFIG = {
    "scenario": "transport",
    "num_envs": 32,             # Number of parallel environments
    "device": "cpu",            # Computing device
    "continuous_actions": True, # Continuous action space
    "max_steps": 500,           # Maximum steps
    "n_agents": 4,              # Number of agents
    "n_packages": 1,            # Number of packages
    "package_width": 0.15,      # Package width
    "package_length": 0.15,     # Package length
    "package_mass": 50,         # Package mass
}
```

5.2 Training Configuration

```
TRAINING_CONFIG = {
    "lr": 3e-4,                 # Learning rate
    "gamma": 0.99,              # Discount factor
    "lambda_": 0.95,            # GAE parameter
    "clip_param": 0.2,          # PPO clipping parameter
    "vf_loss_coeff": 0.5,       # Value function loss coefficient
    "entropy_coeff": 0.01,      # Entropy coefficient
    "ppo_epochs": 10,           # Number of PPO update epochs
    "batch_size": 64,           # Batch size
}
```

6 Experimental Results

6.1 Training Performance

Table 1: Training Performance Comparison

Algorithm	Training Time	Final Avg Reward	Peak Avg Reward	Stability
CPPO	782.93s	-0.1356	0.3457	Medium
MAPPO	815.16s	0.0381	0.2976	Good
IPPO	788.11s	-0.0477	0.1458	Poor

6.2 Result Consistency Analysis

Experimental results verify the paper's conclusions in terms of core performance trends. Specifically:

Peak Performance: CPPO achieved the highest average reward (0.3457) among all algorithms during early training, which aligns with the paper's view that centralized training can achieve theoretically optimal performance.

Algorithm Ranking: In terms of peak performance, the ranking CPPO > MAPPO > IPPO emerged, validating the advantage of centralized training in collaborative tasks.

Stability Difference: Although MAPPO's final convergence value (0.0381) is lower than CPPO's peak, it demonstrates superior stability. In contrast, IPPO performed the worst throughout due to lack of global information, which is completely consistent with expectations.

7 Algorithm Improvement

7.1 Improvement Background

In the original reproduction experiments, we identified the following key issues:

1. **CPPO Insufficient Stability**: Although peak performance was highest (0.3457), it fluctuated violently in later stages, with final reward dropping to negative value (-0.1356) 2. **MAPPO Moderate Convergence Speed**: Required relatively long time to achieve good performance 3. **IPPO Poor Collaboration**: Worst performance due to lack of global information and communication mechanisms

7.2 Observation Normalization Implementation

Problem Root Cause:

- VMAS is based on physics engine, observations contain physical quantities of different scales
- Position range [-1,1], velocity range [-10,10], scale difference reaches 10x
- This scale difference leads to training instability

Solution:

- Implement running mean and variance calculation
- Use normalization formula to convert observations to standard normal distribution
- Clip to reasonable range to prevent extreme values

7.3 Improvement Experimental Results

Table 2: MAPPO Algorithm Performance Comparison (300 iterations)

Metric	Original MAPPO	Improved MAPPO (Observation Normalization)	Improve
Final Reward	-0.1255	**0.0284**	**+0.1540 (-**
Peak Reward	0.1612	**0.6049**	**+0.4436 (-**
Average Reward	-0.0830	-0.0234	+0.0597 (-
Training Time	794.54s	795.96s	+1.41s (-

7.4 Improvement Mechanism Analysis

Gradient Balancing Mechanism:

- Before normalization: Velocity dimension ([-10,10]) is 10x larger than position dimension ([-1,1])
- After normalization: All dimensions are in [-10,10] range
- Effect: Gradient updates are balanced, network can learn all dimensions simultaneously

Optimization Space Improvement:

- Before normalization: Input scale inconsistency causes distorted optimization space
- After normalization: Input approaches standard normal distribution
- Effect: Optimization space is more regular, gradient descent is more effective

8 Conclusion

8.1 Main Achievements

1. **VMAS Code Annotation**: Added detailed Chinese annotations to VMAS core code, covering 9 key files 2. **MARL Algorithm Reproduction**: Successfully reproduced three MARL algorithms (CPPO, MAPPO, IPPO), verified paper conclusions 3. **Algorithm Improvement**: Proposed and implemented observation normalization improvement scheme, with significant performance improvement

8.2 Core Data

Table 3: Core Experimental Data

Metric	Value
MAPPO Final Reward Improvement	+122.6%
MAPPO Peak Reward Improvement	+275.1%
Training Overhead Increase	+0.2%

8.3 Academic Value

1. **Verified VMAS Paper Core Conclusions**: Validated the advantage of centralized training in collaborative tasks 2. **Revealed Impact of Input Scale on MARL**: Identified the problem of input scale differences in physical simulation environments 3. **Proposed Practical Improvement Scheme**: Provided simple and effective improvement strategy applicable to all MARL algorithms

8.4 Practical Application Value

1. **Significantly Improved Algorithm Practicality**: MAPPO final reward improved from negative to positive 2. **Provided Reproducible Improvement Scheme**: Detailed implementation code and complete experimental verification 3. **Lowered MARL Application Threshold**: Simple implementation, strong universality, low risk

9 References

1. Bettini, M., et al. "VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning." arXiv preprint arXiv:2207.03530 (2022).

2. Schulman, J., et al. "Proximal Policy Optimization Algorithms." arXiv preprint arXiv:1707.06347 (2017).

3. Yu, C., et al. "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games." arXiv preprint arXiv:2103.01955 (2021).

4. VMAS GitHub Repository: https://github.com/proroklab/VectorizedMultiAgentSimulator

5. Ray RLlib Documentation: https://docs.ray.io/en/releases-2.6.3/rllib/