CORL2025 - ContinuousMAPF Bench

Anonymous Author(s)
Affiliation
Address
email

Abstract: Multi-agent reinforcement learning (MARL) has become a powerful paradigm capable of coordinated decision-making in both cooperative and competitive multi-agent problems. Although many MARL benchmarks exist, few capture both the complexity and state continuity. In this work, we present CAMAR - a benchmark for continuous multi-agent pathfinding that is specifically designed for ???. The environment simulates planning challenges in continuous state and action spaces, accommodates cooperative and competitive interactions among agents, and scales efficiently to large agent populations with the speed 100K+ environment steps per second. A rich dataset of problem instances and an automated benchmarking tool are provided to facilitate reproducible and fair comparisons across diverse MARL approaches. Experimental results demonstrate that our benchmark delivers a more challenging and representative testbed for advancing MARL research.

Keywords: MARL, MAPF

1 Introduction

Given increasing attention to Multi-agent reinforcement learning (MARL), many researchers have recently focused on complex navigation and coordination challenges in real-world settings. MARL methods have already demonstrated their potential to solve complex tasks in various environments such as ... []. And while a lot of benchmarks have been developed, many of them focus on either discrete environments, where agents operate on fixed grids with predetermined actions, or on continuous settings with relatively simple tasks. As practical applications in robotics require agents to plan smooth trajectories in a continuous space while managing complex interactions.

We have determined that existing benchmarks and simulation environments for MARL are constrained in three ways. First, many rely on discrete representations, which fail to capture the continuous dynamics critical for real-world applications. Second, there are several benchmarks that accurately simulate complex, continuous environments but struggle to scale effectively as the number of agents or objects increases. Third, some environments, [like SMAC], offer continuous state and action spaces and scale to a large number of agents, but they feature simplistic tasks that do not challenge the agents with the complexity found in practical settings.

This lack of a comprehensive platform that is scalable, continuous, and capable of supporting intricate tasks highlights the pressing need for benchmark platforms that embrace continuous state and action spaces, can handle high agent densities, and are specifically tailored to the demands of large-scale MARL.

Contributions To bridge the gap between multi-robot systems and multi-agent reinforcement learning (MARL) research, we introduce the CAMAR (**C**ontinuous **A**ction **M**ulti-**A**gent **R**outing) Benchmark, which offers:

High-Performance Environment: CAMAR is an extremely fast environment with GPU support, achieving speed exceeding 100,000 steps per second. It is specifically designed

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

for multi-agent navigation and collision avoidance tasks with continuous state and action spaces.

Standard Evaluation Protocols: We provide evaluation protocols that include train and holdout task instances, enabling proper assessment of agents' generalization capabilities.

Strong Baselines: For benchmarking purposes, we provide several baselines ranging from both state-of-the-art MARL algorithms and classical solvers.

Figure 1: Env image

2 Related Work

In recent years, a lot of MARL benchmarks have arisen, each addressing various aspects of multi-agent tasks: coordination, planning, navigation, etc. The agents' abilities to plan and navigate in complex conditions and automatically adapt to different environmental changes are crucial for almost every robotic application. Recent benchmarks and environments have already focused on these features; yet they significantly differ in terms of support for continuous observation and action spaces, GPU acceleration, etc. We have evaluated robotic-related multi-agent environments in Table 1 according to the criteria described below.

Continuous Observations and Actions Since robots usually operate in continuous spaces in terms of observations and even actions, it is important to have a benchmark simulating such a scenario. Despite the significant amount of continuous environments, many of them don't focus on robotics tasks (navigation or planning)

GPU Support In multi-agent environments, each agent maintains its own set of observations and rewards, leading to potentially intensive CPU-GPU data transfers—especially in deep RL pipelines. Efficient GPU support is essential for handling such large-scale data processing without becoming a computational bottleneck.

Scalability >500 Agents As the number of agents in an environment increases, so does the complexity of decision-making. Scalability is therefore a key challenge, and our benchmark is designed to efficiently manage environments with over 500 agents, enabling extensive research into large-scale multi-agent systems.

Partially Observable Modern RL commonly operates under conditions of partial observability, where an agent only has access to a limited view of the environment. This feature is crucial for replicating real-world scenarios where full state information is rarely available.

Heterogeneous agents Real-world applications often involve agents with diverse characteristics — such as different sensor types or hardware capabilities. By incorporating heterogeneous agents, our benchmark provides a realistic testbed for evaluating algorithms that must adapt to varied agent behaviors.

Perfomance > 10K Steps/s High simulation speeds are vital for training robust policies. With the need for millions to tens of millions of steps, achieving a throughput of at least 10K steps per second ensures that even simple tasks can be learned in a reasonable timeframe, significantly accelerating experimental research.

Python based A pure Python implementation simplifies usage, support, and customization. This ease of modification is essential for researchers who wish to tailor the environment to their specific experimental needs without the overhead of complex dependencies.

Procedural Generation Procedural generation is widely adopted to expose agents to a diverse range of scenarios. This is critical in multi-agent settings, as it promotes generalization by preventing overfitting to a limited set of predefined configurations.

Requires generalization

Evaluation protocols Robust evaluation protocols are essential for reliably comparing different MARL and MAPF approaches. By standardizing metrics and test scenarios, our benchmark facilitates reproducible and meaningful comparisons of algorithmic performance.

Tests & CI A CI pipeline and testing suite are vital for collaborative development and maintenance of open-source projects. These practices help ensure code reliability and facilitate contributions from the broader research community.

PyPi Listed Availability on PyPi enhances the benchmark's accessibility, allowing researchers to install and integrate it easily into their projects. This ease of deployment fosters rapid experimentation and widespread adoption within the community.

Table 1: Comparison of different MARL environments focusing on planning and navigation

Environment	Repository	Continuous Observations	Continuous Actions	GPU Support	Scalability >500 Agents	Partially observable	Heterogeneous agents	Performance >10K Steps/s	Python based	Procedural generation	Requires generalization	Evaluation protocols	Tests & CI	PyPi Listed	Task Complexity
Waterworld (SISL) []	link	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✓	▮
RWare (original) []	link	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	▮▮
RWare (Jumanji) []	link	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	▮▮
RWare (Pufferlib) []	link	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	▮▮
Trash Pickup (Pufferlib) []	link	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	▮▮
SMAC (original) []	link	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	▮
SMACv2 (original) []	link	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	▮▮
SMAX (JaxMARL) []	link	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✓	✓	▮▮
JaxNav (JaxMARL) []	link	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓	▮▮
POGEMA (original) []	link	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	▮▮▮
VMAS* (original) []	link	✓	✓	✓	✗	✓	✓	✗ / ✓*	✓	✗	✗ / ✓*	✗	✓	✓	▮▮ / ▮▮▮*
SMART (original) []	link	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	▮▮▮
(Ours)	link	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	▮▮▮

▮- Low ▮▮- Medium ▮▮▮- High

Our benchmark addresses several gaps in MARL research while inspiring by existing frameworks. Many multi-agent reinforcement learning benchmarks have been developed to explore different aspects of multi-agent coordination, planning, and navigation. Notable examples include SMAC, Jumanji, POGEMA, MPE, VMAS, etc., each of them has contributed significantly to the field by introducing various challenges.

SMAC, for instance, has become a widely adopted benchmark for evaluating MARL algorithms in strategic settings. However, it operates in a discrete action space and lacks performance for large-scale experiments. Jumanji, developed to provide flexible and scalable MARL challenges, incorporates procedural generation and GPU acceleration but does not primarily focus on planning-oriented and navigation tasks. POGEMA, on the other hand, is particularly well-suited for pathfinding and grid-based multi-agent navigation, using procedural generation and high agent scalability, though it lacks continuous observation-action spaces required for many robotic applications.

The Multi-Agent Particle Environment (MPE) has played a foundational role in MARL research, offering simple and diverse multi-agent scenarios. However, MPE was not initially designed to han-

dle large-scale multi-agent settings with hundreds or even thousands of agents. VMAS extends the ideas of MPE by integrating MPE physics, which allows for more realistic robotic dynamics. While VMAS significantly improves upon MPE, its computational efficiency can still be a bottleneck, especially when scaling to hundreds of agents and obstacles.

Additionally, evaluation protocols are often absent or inconsistently defined across many benchmarks, making cross-method comparisons difficult and limiting the reproducibility of MARL research. Some environments also suffer from performance bottlenecks due to CPU-GPU communication inefficiencies, limiting their applicability to large-scale reinforcement learning.

CAMAR Environment Overview

CAMAR is designed for continuous-space planning tasks in multi-agent settings, where multiple agents must navigate toward target destinations while avoiding static obstacles and collisions with one another. Operating in a fully continuous, vectorized 2D environment, agents use physical forces as actions to move dynamically.

Dynamic

Similar to environments such as MPE and VMAS, CAMAR integrates agents' forces and velocities, applies collision repulsion, and updates both states and observations at every timestep using a (semi-implicit) Euler method:

$$\begin{cases} f_i(t) = f_i^a(t) + \sum_j [f_{ij}^{\text{collision}}(t)] \\ v_i(t+1) = v_i(t) + \frac{\mathbf{f}_i(t)}{m} dt \\ x_i(t+1) = x_i(t) + v_i(t) dt \end{cases} \tag{1}$$

Where $f_{ij}^{\text{collision}}(t) = f_0 \frac{\Delta x_{ij}}{\|\Delta x_{ij}\|} k \log\left(1 + e^{\frac{-(\|\Delta x_{ij}\| - (r_i + r_j))}{k}}\right)$, if $\|\Delta x_{ij}\| < (r_i + r_j)$ and $f_{ij}^{\text{collision}}(t) = 0$ otherwise. This collision repulsion formulation is consistent with the approaches used in MPE and VMAS..

Maps

CAMAR offers a variety of map options that can be customized or replaced with user-defined generators (even if they are non-jittable in JAX). Currently, the following map types are available:

- **RandomGrid**. A procedurally generated grid map with obstacles, agents, and goals randomly placed.
- **StringGrid**. A grid map defined by a string-based layout. Obstacles are placed according to the input string (as in benchmarks like MovingAI), while agent and goal positions can be either predefined or randomized.
- **BatchedStringGrid**. An extension of StringGrid that supports multiple obstacle configurations across batched parallel environments.
- **FluctCont**. A continuous map with a landscape defined by trigonometric functions.
- **PerlinCont**. A continuous map where the landscape is generated using the Perlin noise similarly to video games.

Observations

Each agent obtains a local, ego-centric observation defined by a user-specified window size. The observation mechanism is inspired by LIDAR sensors but avoids the computational complexity of ray tracing by providing a vectorized representation of the surrounding area.

Reward

CAMAR adopts reward formulations common in path-finding environments. Two reward options are available:

- **Sparse Reward**. Agents receive a positive reward when near their goal positions and a negative reward upon colliding with obstacles or other agents.
- **Dense Reward**. In addition to the sparse reward signals, agents receive additional reward feedback proportional to their distance to the goal.

Metrics

Integrations

CAMAR conforms to the Gymnax interface, ensuring familiarity and ease-of-use for researchers. Furthermore, we provide an environment wrapper for TorchRL and seamless integrations with BenchMARL, enabling quick adoption across modern MARL frameworks.

Scalability and Simulation Speed

* Experiments with the simulation speed *

Benchmarks

* results from BecnhMARL *

References