CAMAR: Continuous Actions Multi-Agent Routing

Artem Pshenitsyn[1,2], **Aleksandr Panov**[1,2], **Alexey Skrynnik**[1,2]
[1] AIRI, Moscow, Russia
[2] MIPT, Moscow, Russia
{pshenitsyn, panov, skrynnik}@airi.net

Abstract

Multi-agent reinforcement learning (MARL) is a powerful paradigm for solving cooperative and competitive decision-making problems. While many MARL benchmarks have been proposed, few combine continuous state and action spaces with challenging coordination and planning tasks. We introduce CAMAR, a new MARL benchmark designed explicitly for multi-agent pathfinding in environments with continuous actions. CAMAR supports cooperative and competitive interactions between agents and runs efficiently at up to 100,000 environment steps per second. We also propose a three-tier evaluation protocol to better track algorithmic progress and enable deeper analysis of performance. In addition, CAMAR allows the integration of classical planning methods such as RRT and RRT* into MARL pipelines. We use them as standalone baselines and combine RRT* with popular MARL algorithms to create hybrid approaches. We provide a suite of test scenarios and benchmarking tools to ensure reproducibility and fair comparison. Experiments show that CAMAR presents a challenging and realistic testbed for the MARL community.

Code — https://github.com/AIRI-Institute/CAMAR.git

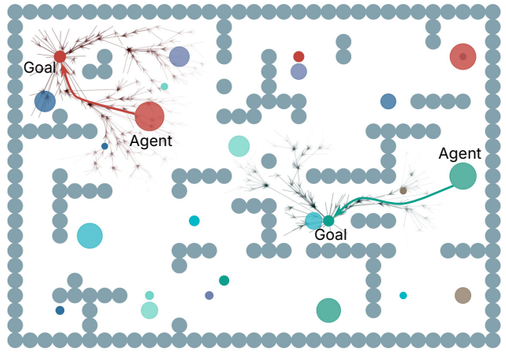


Figure 1: An example scenario from the proposed CAMAR benchmark. Agents are represented as filled circles. Each agent aims to reach its goal while avoiding collisions. The small arrows for the red and green agents indicate segments of paths generated by RRT*, providing guidance for the RL algorithms.

Introduction

Multi-agent reinforcement learning (MARL) has shown strong results in cooperative and competitive settings. Many recent studies explore how MARL can solve various tasks, including navigation and coordination, even in complex environments [1, 2, 3, 4, 5, 6]. However, current MARL benchmarks do not fully match the needs of real-world applications such as robotics. Among the many MARL scenarios, navigation and path planning are especially important for robotic systems. Yet, most existing benchmarks simplify navigation into grid worlds and discrete actions that fail to capture the smooth motion and collision avoidance that robots require.

In addition, learnable methods have recently become the norm for pathfinding tasks because they can handle many agents at once [7, 5, 6, 4]. At the same time, non-learnable approaches struggle with computation as the number of agents grows. Therefore, this work focuses on learnable approaches for path finding in continuous multi-agent settings.

Robots move in continuous space and follow real physics, so they need to plan smooth paths that avoid walls and other robots simultaneously. The task of multi-robot navigation and avoiding collisions with dynamic obstacles is fundamental in robotics and has been studied for a long time [8, 9, 7, 10]. At the same time, most of the work uses its own specific environments and tasks, primarily built using slow simulators such as Gazebo[1] and Isaac Sim[2]. To do this well, we need environments that combine continuous movement with multi-agent interaction. These environments must run fast, use realistic physics, and support many agents learning to navigate without collisions. High speed lets researchers run many experiments and test methods quickly. They should also include built-in benchmarks and evaluation tasks so researchers can compare different methods under the same conditions easily.

We have identified three main limitations in existing MARL environments. First, many use discrete environments, which cannot represent the smooth movements and physical dynamics important in real applications [11, 12, 13]. Second, while some benchmarks do support continuous environments

[1]https://gazebosim.org/home
[2]https://developer.nvidia.com/isaac/sim

arXiv:2508.12845v1 [cs.AI] 18 Aug 2025

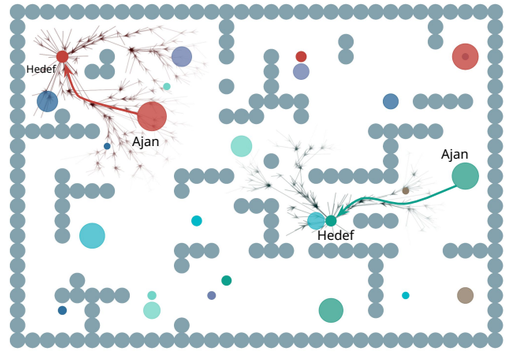
CAMAR: Sürekli Eylemler Çok Etmenli Yönlendirme

Artem Pshenitsyn[1,2], **Aleksandr Panov**[1,2], **Alexey Skrynnik**[1,2]
[1] AIRI, Moskova, Rusya
[2] MIPT, Moskova, Rusya
{pshenitsyn, panov, skrynnik}@airi.net

Özet

Çok etmenli takviyeli öğrenme (ÇEÖ), işbirlikçi ve rekabetçi karar verme problemlerini çözmek için güçlü bir paradigmadır. Birçok ÇEÖ karşılaştırma ölçütü önerilmiş olmasına rağmen, az sayıda ölçüt sürekli durum ve eylem uzaylarını zorlu koordinasyon ve planlama görevleriyle birleştirmektedir. Sürekli eylemlere sahip ortamlarda çok etmenli yol bulma için açıkça tasarlanmış yeni bir ÇEÖ karşılaştırma ölçütü olan CAMAR'ı tanıtıyoruz. CAMAR, ajanlar arasında işbirlikçi ve rekabetçi etkileşimleri destekler ve saniyede 100.000 ortam adımına kadar verimli bir şekilde çalışır. Algoritmik ilerlemeyi daha iyi izlemek ve performansın daha derin analizini sağlamak için üç aşamalı bir değerlendirme protokolü de öneriyoruz. Ek olarak, CAMAR klasik planlama yöntemleri olan RRT ve RRT*'nin MARL süreçlerine entegrasyonuna imkan tanımaktadır. Bunları bağımsız karşılaştırma ölçütleri olarak kullanmakta ve RRT*'yi popüler MARL algoritmalarıyla birleştirerek karma yaklaşımlar geliştirmekteyiz. Yeniden üretilebilirlik ve adil karşılaştırma sağlamak amacıyla kapsamlı test senaryoları ve karşılaştırma ölçütü araçları sağlamaktayız. Deneyler, CAMAR'ın MARL topluluğu için zorlu ve gerçekçi bir test ortamı sunduğunu ortaya koymaktadır.

Kod— https://github.com/AIRI-Institute/CAMAR.git



Şekil 1: Önerilen CAMAR karşılaştırma ölçütünden bir örnek senaryo. Ajanlar dolu çemberler ile temsil edilmiştir. Her ajan, çarpışmalardan kaçınarak hedefine ulaşmayı amaçlamaktadır. Kırmızı ve yeşil ajanlar için küçük oklar, RRT* tarafından oluşturulan yol segmentlerini göstermekte olup pekiştirmeli öğrenme algoritmaları için rehberlik sağlamaktadır.

Giriş

Çok ajanlı takviyeli öğrenme (MARL), iş birliği ve rekabet ortamlarında güçlü sonuçlar vermiştir. Son dönem birçok çalışma, MARL'nin navigasyon ve koordinasyon gibi çeşitli görevleri karmaşık ortamlarda çözme potansiyelini araştırmaktadır [1, 2, 3, 4, 5, 6]. Ancak mevcut MARL karşılaştırma ölçütleri, robotik gibi gerçek dünya uygulamalarının gereksinimlerini tam olarak karşılamamaktadır. Çok sayıda MARL senaryosu arasında, navigasyon ve yol planlama robotik sistemler açısından özellikle önem taşımaktadır. Buna karşın mevcut karşılaştırma ölçütlerinin çoğu navigasyonu, robotların gerektirdiği akıcı hareket ve çarpışmadan kaçınmayı yansıtamayan ızgara tabanlı dünyalar ve ayrık eylemlerle sınırlandırmaktadır.

Ayrıca, öğrenilebilir yöntemler çok sayıda ajanın aynı anda işlenebilmesi nedeniyle yol bulma görevlerinde yakın zamanda standart hale gelmiştir [7,5,6,4]. Buna karşın, öğrenilemeyen yaklaşımlar ajan sayısı arttıkça hesaplama açısından zorlanmaktadır. Bu nedenle, bu çalışma sürekli çok ajanlı ortamlarda yol bulma için öğrenilebilir yaklaşımlara odaklanmaktadır.

Robotlar sürekli bir uzayda hareket eder ve gerçek fiziğe tabidir; bu nedenle, duvarlar ve diğer robotlarla çarpışmadan kaçınan düzgün yollar planlamaları gerekmektedir. Çoklu robot navigasyonu ve dinamik engellerle çarpışmalardan kaçınma görevi temel bir konudur Robotikte uzun zamandır incelenen temel bir konudur [8,9,7,10]. Aynı zamanda, çoğu çalışma kendi özgün ortam ve görevlerini kullanmakta olup öncelikle Gazebo [1] ve Isaac Sim [2] gibi yavaş simülatörler kullanılarak inşa edilmiştir. Bunu iyi gerçekleştirebilmek için sürekli hareketle çok ajanlı etkileşimi birleştiren ortamlar gereklidir. Bu ortamlar hızlı çalışmalı, gerçekçi fizik kullanmalı ve birçok ajanın çarpışma olmadan gezinmeyi öğrenmesini desteklemelidir. Yüksek hız, araştırmacıların çok sayıda deney yapmalarına ve yöntemleri hızlıca test etmelerine olanak tanır. Ayrıca, entegre karşılaştırma ölçütleri ve değerlendirme görevleri içermelidir; böylece araştırmacılar farklı yöntemleri aynı koşullar altında kolayca karşılaştırabilir.

Mevcut MARL ortamlarında üç temel sınırlama tespit ettik . Birincisi, birçok ortam ayrık yapılar kullanmakta olup, bu da gerçek uygulamalarda önemli olan pürüzsüz hareketleri ve fiziksel dinamikleri temsil edemez [11,12,13]. İkincisi, bazı karşılaştırma ölçütleri sürekli ortamları desteklese de

[1]https://gazebosim.org/home
[2]https://developer.nvidia.com/isaac/sim

and complex tasks, they do not scale well when the number of agents and obstacles becomes large [14]. Third, some environments can scale and use continuous states and actions, but their tasks are too simple and do not challenge agents to develop advanced policies [15, 16, 17].

Because of these limitations, a new MARL benchmark for robotic navigation tasks that combines three key features is clearly needed: It should support continuous state and action spaces, handle hundreds or thousands of agents, and include tasks complex enough to reflect the challenges of real-world problems.

To bridge the gap between multi-robot systems and MARL research, we introduce the CAMAR (Continuous Actions Multi-Agent Routing) Benchmark. Specifically, we make the following contributions:

- We introduce CAMAR, an extremely fast environment with GPU acceleration support (using JAX), achieving speeds exceeding 100,000 steps per second. It is designed for multi-agent navigation and collision avoidance tasks in continuous state and action spaces.
- We propose an evaluation protocol that includes both training and holdout task instances, as well as a suite of metrics and performance indicators to assess agents' generalization capabilities.
- We provide strong baselines for benchmarking, including state-of-the-art MARL algorithms and classical path planning methods commonly used in robotics, and conduct an extensive experimental study to evaluate their performance across diverse scenarios.

Related Work

Many multi-agent reinforcement learning (MARL) benchmarks have been developed in recent years, each focusing on different challenges such as coordination, navigation, or multi-agent planning. A key ability for agents in robotics is to move and adapt in complex environments. Some recent environments explore these abilities, but they differ widely in important properties.

These differences include whether the environment supports continuous spaces, GPU acceleration, and high simulation throughput. Some platforms support only discrete actions, while others offer more realistic continuous control. Environments also vary in their ability to handle large groups of agents or support partial observability.

Other practical features also matter for researchers. Some environments are fully implemented in Python and support flexible task generation, which makes them easier to extend and use in large experiments. Others may lack documentation or require heavy customization.

Finally, only a few MARL environments offer standard evaluation protocols, automated tests, or package installation via PyPi. These features are useful for ensuring fair comparisons between methods, improving reproducibility, and making environments easier to maintain. In our work, we aim to address these points by extending existing protocols and providing a high-performance, flexible platform.

To sum up, our benchmark builds on the strengths of existing environments and aims to complement ongoing efforts in MARL research. Over the years, a range of benchmarks has helped researchers tackle challenges in coordination, planning, and navigation. Well-known examples include SMAC [18, 15, 17], Jumanji [19], POGEMA [12, 13], MPE [20, 16], and VMAS, each of which introduces useful features for different kinds of tasks.

SMAC [18, 15, 17] is popular for testing strategic decision-making, but it uses discrete actions and does not scale well for large environments. Jumanji [19] supports GPU acceleration and procedural generation, but its focus is not on navigation or planning. POGEMA [12, 13] is strong in grid-based navigation and procedural tasks with many agents, but it does not use continuous states or actions, which are important in robotics.

MPE [20, 16] has played an important role in early MARL research, but it cannot scale to hundreds of agents efficiently. VMAS [14] builds on MPE [20, 16] by adding its physics and continuous dynamics, making it better for robotics. However, it can still be slow and difficult to scale to larger agent populations or complex maps.

Many environments also lack evaluation protocols. This makes it difficult to compare algorithms fairly and limits the reproducibility of results. Performance issues also appear in environments that depend heavily on CPU-GPU communication, which slows down training and makes large-scale experiments harder.

Finally, popular robotics simulators such as Gazebo [21], Webots [22] and ARGoS [23] offer continuous observations and actions with realistic physics, but they do not provide the ability for fast training in multi-agent scenarios. This gap shows the need for a new simulation environment that brings together high performance and support for large-scale multi-agent navigation and planning. For more detailed analysis of each environment see Appendix H.

Continuous Observations and Actions Robots usually operate in continuous state and action spaces, so it's important for benchmarks to reflect these conditions. Environments like VMAS and MPE [20, 16] use continuous actions and observations, while others, like Trash Pickup [25] or POGEMA [13], use discrete representations, which limit realism in robotics simulations.

GPU Support In multi-agent environments, each agent often has its observations and rewards. This can lead to heavy data transfers between the CPU and the GPU, especially when training deep RL models. Benchmarks like Jumanji [19] and SMAX [17] are initially built with GPU acceleration, helping to reduce training time. In contrast, many older environments like SMAC [18, 15] and MPE [20, 16] do not support GPU-based simulation.

Scalability $>$500 Agents When the number of agents grows, decision-making becomes harder. A good benchmark should scale well to hundreds or thousands of agents. Environments like POGEMA [12, 13] and Trash Pickup [25] can run with thousands and even millions of agents. Others, like VMAS, MPE [20, 16], and SMAC [18, 15], are limited to much smaller groups.

ve karmaşık görevlerde, ajan sayısı ve engeller arttığında iyi ölçeklenmezler [14]. Üçüncü olarak, bazı ortamlar sürekli durumlar ve eylemlerle ölçeklenebilir, ancak görevleri çok basit olup ajanların gelişmiş politikalar geliştirmesini zorlayacak seviyede değildir [15, 16, 17].

Bu sınırlamalar nedeniyle, üç temel özelliği birleştiren robotik navigasyon görevleri için yeni bir MARL karşılaştırma ölçütüne açıkça ihtiyaç vardır: Sürekli durum ve eylem alanlarını desteklemeli, yüzlerce veya binlerce ajanı yönetebilmeli ve gerçek dünya problemlerinin zorluklarını yansıtacak kadar karmaşık görevler içermelidir.

Çoklu robot sistemleri ile MARL araştırmaları arasındaki boşluğu kapatmak amacıyla CAMAR (Continuous Actions Multi-Agent Routing) Karşılaştırma Ölçütü'nü sunuyoruz. Özellikle, aşağıdaki katkılarda bulunuyoruz:

- CAMAR'ı tanıtıyoruz; JAX kullanarak GPU hızlandırması destekli, saniyede 100.000 adımdan fazla hızlara ulaşan son derece hızlı bir ortamdır. Sürekli durum ve eylem alanlarındaki çoklu ajan navigasyon ve çarpışmadan kaçınma görevleri için tasarlanmıştır.
- Eğitim ve tutma görev örneklerini içeren bir değerlendirme protokolü ile ajanların genelleme yeteneklerini ölçmek için çeşitli metrikler ve performans göstergeleri sunuyoruz.
- Robotikte yaygın kullanılan son teknoloji MARL algoritmaları ve klasik yol planlama yöntemlerini içeren güçlü karşılaştırma ölçütü referansları sağlamakta ve performanslarını çeşitli senaryolarda değerlendirmek amacıyla kapsamlı bir deneysel çalışma yürütmekteyiz.

İlgili Çalışmalar

Son yıllarda, koordinasyon, navigasyon veya çok ajanlı planlama gibi farklı zorluklara odaklanan çok sayıda çok ajanlı takviyeli öğrenme (MARL) karşılaştırma ölçütü geliştirilmiştir . Robotikte ajanların temel yeteneklerinden biri karmaşık ortamlarda hareket etmek ve uyum sağlamaktır. Bazı yeni ortamlar bu yetenekleri incelemekte, ancak önemli özellikler bakımından büyük farklılıklar göstermektedir.

Bu farklar, ortamın sürekli alanları destekleyip desteklememesi, GPU hızlandırması ve yüksek simülasyon verimliliği gibi özellikleri kapsamaktadır. Bazı platformlar yalnızca ayrık eylemleri desteklerken, diğerleri daha gerçekçi sürekli kontrol sunmaktadır.

Ortamlar ayrıca büyük ajan gruplarını yönetebilme veya kısmi gözlemlenebilirlik desteği sağlama kapasitesi açısından da farklılık göstermektedir.

Araştırmacılar için diğer pratik özellikler de önem taşımaktadır. Bazı ortamlar tamamen Python ile uygulanmış olup esnek görev oluşturmayı desteklemekte, bu da onları büyük ölçekli deneylerde kullanımı ve genişletilmesini kolaylaştırmaktadır. Diğer ortamlar ise dokümantasyon eksikliği yaşayabilir veya yoğun özelleştirme gerektirebilir.

Son olarak, yalnızca az sayıda MARL ortamı standart değerlendirme protokolleri, otomatik testler veya PyPi üzerinden paket kurulumu sunmaktadır. Bu özellikler, yöntemler arasında adil karşılaştırmalar yapılmasını sağlamak, tekrarlanabilirliği artırmak ve ortamların bakımını kolaylaştırmak açısından önemlidir. Çalışmamızda , mevcut protokolleri genişleterek ve yüksek performanslı, esnek bir platform sunarak bu hususlara çözüm getirmeyi hedefliyoruz.

Özetle, karşılaştırma ölçütümüz mevcut ortamların güçlü yönleri üzerine inşa edilmiş olup, devam eden MARL araştırmalarını tamamlamayı amaçlamaktadır.

çabalarında. Zaman içinde, çeşitli karşılaştırma ölçütleri koordinasyon, planlama ve navigasyon alanlarındaki zorlukların üstesinden gelinmesinde araştırmacılara destek olmuştur. Yaygın bilinen örnekler arasında SMAC [18,15,17], Jumanji [19], POGEMA [12,13], MPE [20,16] ve VMAS yer almakta olup, her biri farklı görev tipleri için faydalı özellikler sunmaktadır.

SMAC [18,15,17] stratejik karar verme testlerinde yaygın olarak kullanılmaktadır, ancak ayrık eylemler kullanmakta ve büyük ortamlar için ölçeklenmesi zordur. Jumanji [19], GPU hızlandırmasını ve prosedürel üretimi desteklemektedir , ancak odak noktası navigasyon veya planlama değildir. POGEMA [12,13], çok sayıda ajanla ızgara tabanlı navigasyon ve prosedürel görevlerde güçlüdür, ancak robotik için önemli olan sürekli durumlar veya eylemler kullanmamaktadır.

MPE [20,16], erken çok ajanlı pekiştirmeli öğrenme araştırmalarında önemli bir rol oynamış, ancak yüzlerce ajana verimli şekilde ölçeklenememektedir. VMAS [14], fizik ve sürekli dinamikler ekleyerek MPE [20,16] üzerine inşa edilmiş ve robotik için daha uygundur. Ancak, daha büyük ajan popülasyonlarına veya karmaşık haritalara ölçeklenmesi hâlâ yavaş ve zordur.

Pek çok ortamda ayrıca değerlendirme protokolleri bulunmamaktadır. Bu durum, algoritmaların adil karşılaştırılmasını zorlaştırmakta ve sonuçların tekrarlanabilirliğini sınırlamaktadır. Performans sorunları, eğitim sürecini yavaşlatan ve büyük ölçekli deneyleri zorlaştıran CPU-GPU iletişimine yoğun biçimde bağlı ortamlarda da ortaya çıkmaktadır.

Son olarak, Gazebo [21], Webots [22] ve ARGoS [23] gibi popüler robotik simülatörler gerçekçi fizik modelleri ile sürekli gözlem ve eylem sağlamaktadır, ancak çok ajanlı senaryolarda hızlı eğitim olanağı sunmamaktadır. Bu eksiklik , yüksek performans ile büyük ölçekli çok ajanlı navigasyon ve planlamayı destekleyen yeni bir simülasyon ortamı gerekliliğini ortaya koymaktadır. Her ortamın daha ayrıntılı analizi için Ek H'ye bakınız.

Sürekli Gözlemler ve Eylemler Robotlar genellikle sürekli durum ve eylem alanlarında çalıştığından, karşılaştırma ölçütlerinin bu koşulları yansıtması önem arz etmektedir. VMAS ve MPE [20,16] gibi ortamlar sürekli eylem ve gözlemler kullanırken, Çöp Toplama [25] veya POGEMA [13] gibi diğer ortamlar ayrık temsiller kullanmakta olup, bu robotik simülasyonlardaki gerçekçiliği sınırlandırmaktadır.

GPU Desteği Çok ajanlı ortamlarda, her ajan genellikle kendi gözlemlerine ve ödüllerine sahiptir. Bu durum, özellikle derin PE modellerinin eğitiminde, CPU ile GPU arasında yoğun veri aktarımına yol açabilir. Jumanji [19] ve SMAX [17] gibi karşılaştırma ölçütleri başlangıçta GPU hızlandırması ile oluşturulmuş olup, eğitim süresini azaltmaya yardımcı olmaktadır. Buna karşılık, SMAC [18, 15] ve MPE [20, 16] gibi birçok eski ortam GPU tabanlı simülasyonu desteklememektedir.

Ölçeklenebilirlik $>$ **500** Ajan Ajan sayısı arttıkça, karar verme süreci zorlaşmaktadır. İyi bir karşılaştırma ölçütü, yüzlerce veya binlerce ajana kolayca ölçeklenebilmelidir. POGEMA [12, 13] ve Çöp Toplama [25] gibi ortamlar binlerce hatta milyonlarca ajanla çalıştırılabilmektedir. VMAS, MPE [20, 16] ve SMAC [18, 15] gibi diğerleri çok daha küçük gruplarla sınırlıdır.

Environment / Simulator	Repository	Cont. Observations	Cont. Actions	GPU Support	Scalability >500 Agents	Partially observable	Heterogeneous agents	Performance >10K SPS	Python based	Procedural generation	Requires generalization	Evaluation protocols	Tests & CI	PyPI Listed
Waterworld (SISL) [24]	link	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✓
RWare [11]	link	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓
RWare (Jumanji) [19]	link	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓
RWare (Pufferlib) [25]	link	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓
Trash Pickup (Pufferlib) [25]	link	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓
SMAC [18]	link	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗
SMACv2 [15]	link	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗
SMAX (JaxMARL) [17]	link	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✓	✓
MPE [20, 16]	link	✓	✓	✗	✓	✓	✓	✗ / ✓ [3]	✓	✗	✓	✗	✓	✓
MPE (JaxMARL) [17]	link	✓	✓	✓	✓	✓	✓	✗ / ✓ [3]	✓	✗	✓	✗	✓	✓
JaxNav (JaxMARL) [26]	link	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓
Nocturne [27]	link	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗
POGEMA [12]	link	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
VMAS [4] [14]	link	✓	✓	✓	✗	✓	✓	✗ / ✓ [4]	✓	✗	✗ / ✓ [4]	✗	✓	✓
SMART [28]	link	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
Gazebo [21]	link	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
Webots [22]	link	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
ARGoS [23]	link	✓	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
CAMAR (Ours)	link	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of multi-agent reinforcement learning (MARL) environments and simulators. Each row corresponds to a specific environment or a particular implementation of it. The columns indicate key properties, including support for continuous observations and actions, GPU acceleration, scalability beyond 500 agents, partial observability, heterogeneous agents, and whether the simulator can exceed 10K simulation steps per second (SPS). Additional columns specify if the environment is implemented fully in Python, supports procedural generation, requires generalization across different maps or tasks, includes evaluation protocols, and provides built-in tests or continuous integration (CI). The "Repository" column contains links to the official source code for each environment. CAMAR, listed at the bottom, is our proposed environment.

Partially Observable In most real-world scenarios, agents see only part of the environment. This is known as partial observability and is a common feature in almost every RL environment. SMAC [18, 15], RWare [11], and POGEMA [12, 13] support this feature by limiting each agent's view.

Heterogeneous Agents In real life, agents and robots can have different sensors, shapes, or goals. Benchmarks like VMAS and SMAC [18, 15] allow agents to behave differently, making cooperation more complex. Others, like POGEMA [12, 13] or RWare [11], usually involve homogeneous agents.

Performance >10K Steps/s High simulation speed is essential when agents need millions of interactions to learn. For example, the Pufferlib [25] environment can run with a speed of up to 1M steps per second. This allows researchers to train and develop models more quickly. Environments like SMAC [18, 15] or VMAS can be slower, especially when running with many agents.

Python Based Using Python makes it easier for researchers to understand and modify the environment. SMAX [17], VMAS, and POGEMA [12, 13] are implemented entirely in Python. This helps with faster development and integration into learning frameworks.

Procedural Generation Procedural generation helps create diverse tasks that reduce the chance of overfitting. POGEMA [12, 13] and Jumanji [19] use this method to generate complex tasks automatically. Other environments, like RWare [11], use fixed layouts that limit diversity.

Requires Generalization Only a few environments offer separate training and testing tasks, making it difficult to test whether agents generalize well to new situations. POGEMA [12, 13] and SMACv2 [15] include test scenarios that allow researchers to check generalization. Many others use the same tasks during both training and testing.

Evaluation Protocols To compare algorithms fairly, we need well-defined test cases and metrics. Benchmarks like SMACv2 [15] and Jumanji [19] include evaluation protocols to make results reproducible and meaningful. Many other

[3] SPS decreases gracefully with many agents and obstacles.
[4] VMAS is a framework consisting of many different scenarios, while some scenarios run efficiently with a speed exceeding 10K SPS, other, complex ones don't; the same applies for generalization.

Ortam / Simülatör	Depo	Sürekli Gözlemler	Sürekli Eylemler	GPU Desteği	Ölçeklenebilirlik >500 Ajan	Kısmen gözlemlenebilir	Heterojen ajanlar	Performans >10K SPS	Python tabanlı	Prosedürel üretim	Genelleme gerektirir	Değerlendirme protokolleri	Testler & CI	PyPI Listelenmiş
Waterworld (SISL) [24]	bağlantı	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✓
RWare [11]	bağlantı	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓
RWare (Jumanji) [19]	bağlantı	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓
RWare (Pufferlib) [25]	bağlantı	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓
Çöp Toplama (Pufferlib) [25]	bağlantı	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓
SMAC [18]	bağlantı	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗
SMACv2 [15]	bağlantı	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗
SMAX (JaxMARL) [17]	bağlantı	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✓	✓
MPE [20, 16]	bağlantı	✓	✓	✗	✓	✓	✓	✗ / ✓ [3]	✓	✗	✓	✗	✓	✓
MPE (JaxMARL) [17]	bağlantı	✓	✓	✓	✓	✓	✓	✗ / ✓ [3]	✓	✗	✓	✗	✓	✓
JaxNav (JaxMARL) [26]	bağlantı	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓
Nocturne [27]	bağlantı	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗
POGEMA [12]	bağlantı	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
VMAS [4] [14]	bağlantı	✓	✓	✓	✗	✓	✓	✗ / ✓ [4]	✓	✗	✗ / ✓ [4]	✗	✓	✓
SMART [28]	bağlantı	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
Gazebo [21]	bağlantı	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
Webots [22]	bağlantı	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
ARGoS [23]	bağlantı	✓	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
CAMAR (Bizim)	bağlantı	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tablo 1: Çok ajanlı takviyeli öğrenme (MARL) ortamları ve simülatörlerin karşılaştırılması. Her satır belirli bir ortamı veya onun özel bir uygulamasını temsil eder. Sütunlar, sürekli gözlemler ve eylemler desteği, GPU hızlandırması, 500 ajanın üzerindeki ölçeklenebilirlik, kısmi gözlemlenebilirlik, heterojen ajanlar ve simülatörün saniyede 10K simülasyon adımını (SPS) aşabilme durumu gibi temel özellikleri gösterir. Ek sütunlar, ortamın tamamen Python ile uygulanıp uygulanmadığını, prosedürel üretimi desteklemediğini, farklı haritalar veya görevler arasında genelleme gerektirip gerektirmediğini, değerlendirme protokollerini içerip içermediğini ve yerleşik testler veya sürekli entegrasyon (CI) sağlayıp sağlamadığını belirtir. "Repository" sütunu, her bir ortam için resmi kaynak kodu bağlantılarını içermektedir. En altta listelenen CAMAR, önerdiğimiz ortamdır.

Kısmi Gözlemlenebilirlik Gerçek dünya senaryolarının çoğunda ajanlar ortamın yalnızca bir kısmını görür. Bu durum kısmi gözlemlenebilirlik olarak adlandırılır ve hemen hemen her RL ortamında yaygın bir özelliktir. SMAC [18, 15], RWare [11] ve POGEMA [12, 13], her ajanın görüş alanını sınırlandırarak bu özelliği destekler.

Heterojen Ajanlar Gerçek hayatta ajanların ve robotların farklı sensörleri, şekilleri veya hedefleri olabilir. VMAS ve SMAC [18, 15] gibi karşılaştırma ölçütleri, ajanların farklı davranmasına izin vererek iş birliğini daha karmaşık hale getirir. POGEMA [12 , 13] veya RWare [11] gibi diğerleri genellikle homojen ajanları içerir.

Performans > 10K Adım/s Ajanların öğrenebilmesi için milyonlarca etkileşim gerektiğinde yüksek simülasyon hızı esastır. Örneğin, Pufferlib [25] ortamı saniyede 1 milyon adıma kadar çalıştırabilir. Bu durum, araştırmacıların modelleri daha hızlı eğitip geliştirmesine olanak sağlar. SMAC [18, 15] veya VMAS gibi ortamlar, özellikle çok sayıda ajanla çalıştırıldığında daha yavaş olabilir.

Python Tabanlı Python kullanımı, araştırmacıların ortamı anlamasını ve değiştirmesini kolaylaştırır. SMAX [17], VMAS ve POGEMA [12, 13], tamamen Python'da uygulanmıştır. Bu, daha hızlı geliştirme ve öğrenme çerçevelerine entegrasyonu sağlar.

Prosedürel Üretim Prosedürel üretim, aşırı uyumu azaltan çeşitli görevlerin oluşturulmasına yardımcı olur. POGEMA [12, 13] ve Jumanji [19], bu yöntemi karmaşık görevleri otomatik olarak oluşturmak için kullanır. RWare [11] gibi diğer ortamlar, çeşitliliği sınırlandıran sabit düzenler kullanır.

Genelleme Gerektirir Sadece birkaç ortam, ayrı eğitim ve test görevleri sunar; bu da ajanların yeni durumlara iyi genelleyip genellemediğini test etmeyi zorlaştırır. POGEMA [12,13] ve SMACv2 [15], araştırmacıların genelleme yeteneğini kontrol etmelerine olanak tanıyan test senaryolarını içermektedir. Birçok diğer yöntem , hem eğitim hem de test sırasında aynı görevleri kullanmaktadır.

Değerlendirme Protokolleri Algoritmaları adil bir şekilde karşılaştırmak için iyi tanımlanmış test vakalarına ve metriklere ihtiyaç vardır. SMACv2 [15] ve Jumanji [19] gibi karşılaştırma ölçütleri, sonuçların çoğaltılabilir ve anlamlı olmasını sağlamak amacıyla değerlendirme protokolleri içermektedir. Birçok diğer

[3] SPS, çok sayıda ajan ve engel ile kademeli olarak azalmaktadır.
[4] VMAS, birçok farklı senaryodan oluşan bir çerçevedir; bazı senaryolar 10K SPS hızını aşan verimlilikle çalışırken, daha karmaşık olanlar çalışmamaktadır; aynı durum genelleme için de geçerlidir.

environments do not have standard evaluation tools or test setups.

Tests & CI 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 whole research community.

PyPi Listed When an environment is available on PyPi, it becomes easier for others to install and use. Benchmarks like Jumanji [19], and Trash Pickup [25] are available on PyPi. This lowers the barrier to entry and helps increase adoption in the research community.

CAMAR Environment

CAMAR is designed for continuous-space planning tasks in multi-agent environments. In this environment, multiple agents move toward their goals while avoiding both static obstacles and other moving agents. The simulation happens in a fully continuous two-dimensional space, without any predefined grids. Agents interact by applying forces, which control their movement through a simple and computationally efficient dynamic model. This approach makes the environment more realistic and easier to scale for many agents.

Dynamic Model & Action Space A key part of CAMAR is its collision model. Similar to MPE [20, 16] and VMAS [14], CAMAR uses a force-based system. Agents receive repulsive forces from nearby agents and obstacles. The collision force applied to agent i from object j is calculated using a smooth contact model, as shown in the equation below.

$$\begin{cases} \vec{f}_{ij}^{\text{collision}}(t) = f_0 \frac{\Delta\vec{x}_{ij}(t)}{\|\Delta\vec{x}_{ij}(t)\|} k \log\left(1 + e^{\frac{-(\|\Delta\vec{x}_{ij}(t)\| - d_{\min})}{k}}\right), \\ \text{if } \|\Delta\vec{x}_{ij}(t)\| < d_{\min}; \\ \\ \vec{f}_{ij}^{\text{collision}}(t) = 0, \\ \text{otherwise.} \end{cases}$$

Here, contact force f_0 regulates the magnitude of the repulsive force, penetration softness k controls the smoothness of the contact dynamics, $\Delta\vec{x}_{ij}(t)$ is a displacement vector between agent i and an object j at time t, $d_{\min}$ defines the minimum allowable distance before collision is activated.

When two objects overlap, the force grows smoothly without sudden changes. When there is no overlap, the collision force is zero. This smooth behavior helps keep agent movement more realistic and stable. The total collision force acting on agent i is calculated by summing forces from all nearby objects: $\vec{f}i^{c}(t) = \sum j \vec{f}_{ij}^{\text{collision}}(t)$.

The full environment state is updated using the collision force and agents' actions. CAMAR supports multiple types of dynamic models. Currently, we provide two built-in models: `HolonomicDynamic` and `DiffDriveDynamic`.

HolonomicDynamic This model is simple and similar to the one used in MPE [20, 16]. Each agent has a position and velocity. The agent moves by applying a 2D force. The

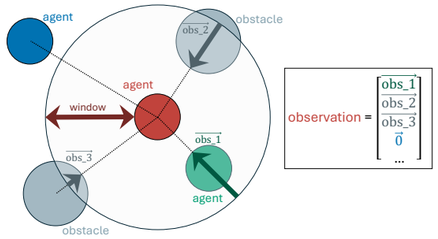


Figure 3: LIDAR-inspired vector observations in CAMAR. Each agent detects nearby objects using penetration vectors, and receives a normalized goal direction.

next state is calculated using the semi-implicit Euler method, as described in the equation below.

$$\begin{cases} \vec{v}_i(t+dt) = (1 - \texttt{damping})\vec{v}_i(t) + \frac{\vec{f}_i^a(t) + \vec{f}_i^c(t)}{\texttt{m}} dt \\ \vec{v}_i(t+dt) := \begin{cases} \vec{v}_i(t+dt), \text{if } \|\vec{v}_i(t+dt)\| < \texttt{max\_speed} \\ \frac{\vec{v}_i(t+dt)}{\|\vec{v}_i(t+dt)\|} \cdot \texttt{max\_speed}, \text{otherwise.} \end{cases} \\ \vec{\text{pos}}_i(t+dt) = \vec{\text{pos}}_i(t) + \vec{v}_i(t+dt)dt \end{cases}$$

Here, $\vec{f}_i^a(t)$ is the 2D action force of agent i, `damping` - scalar in the range $[0, 1)$ that controls velocity decay over time, `m` is the agent mass for applying forces, `max_speed` regulates maximum speed of an agent preserving direction but limiting speed, dt is the time step duration between updates.

DiffDriveDynamic Differential-drive robot model is another built-in dynamic in CAMAR. Each agent has a 2D position $\vec{\text{pos}}_i(t)$ and a heading angle $\theta_i(t)$. The agent chooses a 2D action: one value for linear speed $u_i^a(t)$ and one for angular speed $\omega_i^a(t)$. The motion is updated based on this action using equation below.

$$\begin{cases} u_i(t) = \text{clip}(u_i^a(t), -\texttt{max\_u}, \texttt{max\_u}) \\ \omega_i(t) = \text{clip}(\omega_i^a(t), -\texttt{max\_w}, \texttt{max\_w}) \\ \vec{v}_i(t) = [u_i(t)\cos(\theta_i(t)); u_i(t)\sin(\theta_i(t))] \\ \vec{\text{pos}}_i(t+dt) = \vec{\text{pos}}_i(t) + \vec{v}_i(t)dt \\ \theta_i(t+dt) = \theta_i(t) + \omega_i(t)dt \end{cases}$$

Here, `max_u` and `max_w` are constraints on agent velocities.

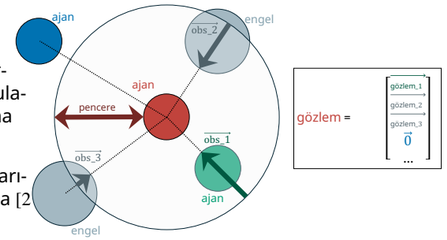
Observations Each agent in CAMAR receives a local observation centered around itself. The size of the observation window can be set by the user. This observation system is inspired by LIDAR sensors but avoids using ray tracing. Instead, CAMAR provides a simple and efficient vector-based observation.

Each agent observes nearby objects using a penetration-based vector representation, which ensures smooth and continuous observations. For every object in the environment

Ortamlarda standart değerlendirme araçları veya test kurulumları bulunmamaktadır.

Testler & CI CI boru hattı ve test birimi, açık kaynak projelerin işbirlikçi geliştirilmesi ve bakımı için hayati önem taşımaktadır. Bu uygulamalar, kod güvenilirliğini sağlamaya yardımcı olur ve tüm araştırma topluluğunun katkılarını kolaylaştırır.

PyPi Listesinde Bir ortam PyPi üzerinde mevcut olduğunda, başkalarının yüklemesi ve kullanması kolaylaşır. Jumanji [19] ve Çöp Toplama [25] gibi karşılaştırma ölçütleri PyPi üzerinde yer almaktadır. Bu durum, giriş engelini düşürür ve araştırma topluluğunda benimsenmeyi artırır.



Şekil 3: CAMAR'da LIDAR uyarlamalı vektör gözlemleri. Her ajan, penetrasyon vektörleri kullanarak yakınındaki nesneleri algılar ve normalleştirilmiş bir hedef yönü alır.

CAMAR Ortamı

CAMAR, çok ajanlı ortamlarda sürekli eylem alanı planlama görevleri için tasarlanmıştır. Bu ortamda, birden fazla ajan hem statik engellerden hem de diğer hareket eden ajanlardan kaçınarak hedeflerine doğru hareket eder. Simülasyon, önceden tanımlanmış ızgaralar olmaksızın tamamen sürekli iki boyutlu bir uzayda gerçekleşir. Ajanlar, hareketlerini kontrol eden kuvvetleri uygulayarak, basit ve hesaplama açısından verimli bir dinamik modelle etkileşimde bulunur. Bu yaklaşım, ortamı daha gerçekçi hale getirir ve çok sayıda ajan için ölçeklenmesini kolaylaştırır.

Dinamik Model & Eylem Alanı CAMAR'ın temel bileşenlerinden biri çarpışma modelidir. MPE [20,16] ve VMAS [14] örneklerinde olduğu gibi, CAMAR da kuvvet-temelli bir sistem kullanmaktadır. Ajanlar, yakınlardaki ajanlar ve engeller tarafından itici kuvvetler alırlar. Ajan i üzerine nesne j tarafından uygulanan çarpışma kuvveti, aşağıdaki denklemde gösterildiği gibi pürüzsüz bir temas modeli kullanılarak hesaplanır.

$$\begin{cases} f_{ij}^{\text{çarpışma}}(t) = f_0 \frac{\Delta x_{ij}(t)}{\|\Delta x_{ij}(t)\|} k \log\left(1 + e^{\frac{-(\|\Delta x_{ij}(t)\| - d_{\min})}{k}}\right), \\ \text{eğer } \|\Delta x_{ij}(t)\| < d_{\min}; \\ \\ f_{ij}^{\text{çarpışma}}(t) = 0, \\ \text{aksi halde.} \end{cases}$$

Burada, temas kuvveti f_0 itici kuvvetin büyüklüğünü düzenler, penetrasyon yumuşaklığı k temas dinamiklerinin pürüzsüzlüğünü kontrol eder, Δ x ij (t) , ajan i ile nesne j arasındaki zaman t deki yer değiştirme vektörüdür, d min çarpışmanın etkinleşmesi için gereken minimum izin verilen mesafeyi tanımlar.

İki nesne çakıştığında, kuvvet ani değişiklikler olmadan düzgün bir şekilde artar. Çakışma olmadığında, çarpışma kuvveti sıfırdır. Bu düzgün davranış, ajan hareketlerinin daha gerçekçi ve stabil kalmasını sağlar. Ajan i üzerinde etkili olan toplam çarpışma kuvveti, tüm yakın nesnelerden gelen kuvvetlerin toplamı olarak hesaplanır: $f i^{c}(t) = \text{sum\_j} \in \text{ec\{f\}\_\{ij\}\^\{çarpışma\}} t)$.

Ortamın tüm durumu, çarpışma kuvveti ve ajanların eylemleri kullanılarak güncellenir. CAMAR, birden çok dinamik model türünü destekler. Şu anda iki yerleşik model sağlanmaktadır: HolonomikDinamikve`DiffDriveDynamic`.

HolonomikDinamik Bu model basit olup, MPE [20, 16]'da kullanılan modele benzemektedir. Her ajanın konumu ve hızı vardır. Ajan, 2B kuvvet uygulayarak hareket eder.

Bir sonraki durum, aşağıdaki denklemde belirtildiği üzere yarı-örtülü Euler yöntemi kullanılarak hesaplanır.

$$\begin{cases} v_i(t+dt) = (1 - \text{damping})v_i(t) + \frac{f_i^a(t) + f_i^c(t)}{\text{m}} dt \\ v_i(t+dt) := \begin{cases} v_i(t+dt), \text{eğer } \|v_i(t+dt)\| < \text{azami hız} \\ \frac{v_i(t+dt)}{\|v_i(t+dt)\|} \cdot \text{azami hız, diğer durumda.} \end{cases} \\ \text{pos}_i(t+dt) = \text{pos}_i(t) + v_i(t+dt)dt \end{cases}$$

Burada, $f_i^a(t)$ ajan i için 2B eylem kuvvetidir, `damping` - zamana bağlı hız azalmasını kontrol eden [0 , 1) aralığında bir skalar, `m` ajanın kuvvet uygulaması için kütlesidir, `max speed` ajanın yönünü koruyarak maksimum hızını sınırlar, dt güncelleme adımı süresidir.

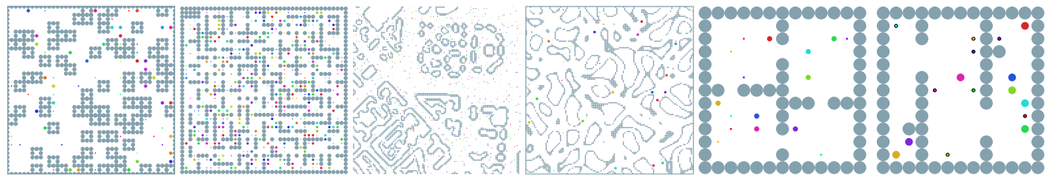
DiffDriveDynamic Diffsiyel sürüş robot modeli, CAMAR'da yerleşik bir diğer dinamiktir. Her ajanın 2B konumu pos $_i$ (t) ve bir yön açısı θ i (t) bulunur. Ajan, doğrusal hız için bir değer u a i (t) ve açısal hız için bir değer ω i a (t) olmak üzere 2B bir eylem seçer. Hareket, aşağıdaki denklem kullanılarak bu eyleme göre güncellenir.

$$\begin{cases} u_i(t) = \text{clip}(u_i^a(t), -\texttt{max\_u}, \texttt{max\_u}) \\ \omega_i(t) = \text{clip}(\omega_i^a(t), -\texttt{max\_w}, \texttt{max\_w}) \\ v_i(t) = [u_i(t)\cos(\theta_i(t)); u_i(t)\sin(\theta_i(t))] \\ \text{pos}_i(t+dt) = \text{pos}_i(t) + v_i(t)dt \\ \theta_i(t+dt) = \theta_i(t) + \omega_i(t)dt \end{cases}$$

Burada,`max u`ve`max w`ajan hızlarına getirilen kısıtlamalardır.

CAMAR'da her ajan, kendisini merkeze alan yerel bir gözlem alır. Gözlem penceresinin boyutu kullanıcı tarafından ayarlanabilir. Bu gözlem sistemi LIDAR sensörlerinden esinlenmiş olup, ışın izleme yöntemini kullanmamaktadır. Bunun yerine, CAMAR basit ve verimli vektör tabanlı bir gözlem sağlar.

Her ajan, pürüzsüz ve sürekli gözlemler sağlamak amacıyla nüfuz tabanlı vektör temsili kullanarak yakınlardaki nesneleri gözlemler. Ortamdaki her nesne için



(a) `random_grid` (b) `labmaze_grid` (c) `movingai` (d) `caves_cont` (e) `string_grid` (f) batched_string_grid

Figure 4: A rich collection of maps for multi-agent planning in continuous spaces in CAMAR: support for both continuous and grid landscapes together with MovingAI collection [29].

(either an agent or a static landmark), the observation is computed as a normalized vector pointing from the agent to the object. If the object is far away, outside the agent's sensing window, the observation becomes a zero vector. This method avoids discontinuities and helps agents better generalize across different object sizes.

$$\begin{cases} \Delta\vec{o}_j = \vec{o}_j - \vec{\text{pos}} \\ \vec{\text{obs}}\_\text{j} = \begin{cases} \Delta\vec{o}_j \cdot \left(1 - \frac{\texttt{window}+R_j}{\|\Delta\vec{o}_j\|}\right), \\ \text{if } \|\Delta\vec{o}_j\| - R - R_j < \texttt{window} \\ \vec{0}, \\ \text{otherwise.} \end{cases} \\ \vec{\text{obs}}\_\text{j} := \frac{\vec{\text{obs}}\_\text{j}}{\texttt{window}} \end{cases}$$

This observation is computed for each agent in a fully vectorized manner. Afterward, only the top `max_obs` closest objects are kept to form the final observation vector. Here, $\vec{o}_j$ is the 2D position of object j, R is the agent radius, R_j the radius of object j, `window` is a parameter that sets how far the agent can sense objects nearby.

In addition to obstacle information, each agent also gets an ego-centric vector pointing to its goal. This vector is clipped, normalized and concatenated to the final observation. This structure helps agents understand both their surroundings and the direction they need to move.

Circle-Based Discretization One simple way to simulate a world is to use geometry rules to check if objects overlap. Ray tracing is a common method for detecting collisions based on the shapes of objects. However, ray tracing is hard to implement in a way that is fast on a GPU. Simple versions of ray tracing are slow and better suited for CPU simulations.

Another method is to discretize the world and only check collisions with nearby objects. Based on the dynamic model described above, it is enough to calculate the distance between objects.

In CAMAR, every object is represented as a circle. This choice has several advantages. Checking the distance between two circles is simple and fast. It does not require special cases like ray tracing does. It also avoids the complexity of handling different shapes like rectangles or polygons. Because of this, the simulation can easily run on GPUs with many agents at the same time. This design allows CAMAR to simulate large-scale multi-agent tasks efficiently and with high performance.

Map Generators Although every object in CAMAR is represented as a circle, it is still possible to create complex and detailed maps. A complex structure can be made by combining many smaller circles close together. The more circles that are used, the more accurate the map becomes. This method allows users to simulate walls, tunnels, mazes, and other complicated shapes even though the basic element is always a circle.

CAMAR includes several types of built-in maps. It also gives users the ability to add custom map generators, even if they are not compatible with JAX just-in-time compilation [30]. A custom map generator can be connected easily by using the `string_grid` or batched_string_grid formats. This flexibility allows users to design many kinds of environments, from simple random setups to complex and realistic maps.

The current set of built-in maps and generators includes (see Fig. 4):

4a `random_grid`: A map where obstacles, agents, and goals are placed randomly on a grid with a predefined size.

4b `labmaze_grid`: Maps generated using LabMaze[5] [31] - maze generator with connected rooms.

4c `movingai`: Integrated two-dimensional maps from the MovingAI benchmark [29], adapted for continuous planning tasks. They can also be used in a `batched_string_grid` manner.

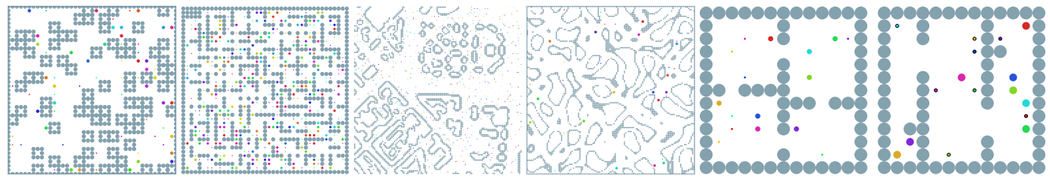
4d `caves_cont`: A continuous type of map where caves are generated using Perlin noise, a common method in video games for creating realistic and varied landscapes.

4e `string_grid`: A grid map based on a text layout. Obstacles are placed according to characters in a string, similar to the MovingAI benchmark [29]. Agent and goal positions can be fixed or random, depending on the free cells in the string.

4f `batched_string_grid`: Similar to `string_grid`, but supports different obstacle layouts across parallel environments. This allows training on multiple map variations at once.

Reward Function CAMAR uses a scalar reward for each agent at every time step. This reward is the sum of four terms: a goal reward, a collision penalty, a movement-based reward, and a collective success reward:

[5] https://github.com/google-deepmind/labmaze



(a) rastgele ızgara (b) labmaze ızgarası (c) `movingai` (d) mağaralar devamı (e) string ızgarası (f) toplu dizge ızgarası

Şekil 4: CAMAR'da sürekli uzaylarda çok ajanlı planlama için zengin bir harita koleksiyonu: hem sürekli hem de ızgara tabanlı ortamların yanı sıra MovingAI koleksiyonu [29] desteği.

(ajan ya da statik bir işaret) olsun, gözlem ajan ile nesne arasına işaret eden normalleştirilmiş bir vektör olarak hesaplanır. Nesne, ajanın algılama penceresinin dışındaysa, gözlem sıfır vektör olur. Bu yöntem süreksizlikleri engeller ve ajanların farklı nesne boyutları arasında daha iyi genelleme yapmalarına olanak tanır.

$$\begin{cases} \Delta\vec{o}_j = \vec{o}_j - \vec{\text{pos}} \\ \vec{\text{obs}}\_\text{j} = \begin{cases} \Delta\vec{o}_j \cdot \left(1 - \frac{\texttt{window}+R_j}{\|\Delta\vec{o}_j\|}\right), \\ \text{if } \|\Delta\vec{o}_j\| - R - R_j < \texttt{window} \\ \vec{0}, \\ \text{otherwise.} \end{cases} \\ \vec{\text{obs}}\_\text{j} := \frac{\vec{\text{obs}}\_\text{j}}{\texttt{window}} \end{cases}$$

Bu gözlem, her ajan için tamamen vektörleştirilmiş şekilde hesaplanır. Daha sonra, yalnızca en yakın `max_obs` nesneler saklanarak nihai gözlem vektörü oluşturulur. Burada, $\vec{o}_j$ j nesnesinin 2B konumu, R ajan yarıçapı, R_j j nesnesinin yarıçapı, window ise ajanın çevresindeki nesneleri algılayabileceği mesafeyi belirleyen bir parametredir.

Engel bilgilerine ek olarak, her ajan hedefini gösteren ego-merkezli bir vektör de alır. Bu vektör kırpılır, normalize edilir ve nihai gözlem vektörüne eklenir. Bu yapı, ajanların hem çevrelerini hem de hareket etmeleri gereken yönü kavramalarına olanak sağlar.

Çember Tabanlı Ayrıklaştırma Bir dünyayı simüle etmenin basit yollarından biri, nesnelerin örtüşüp örtüşmediğini kontrol etmek için geometri kurallarını kullanmaktır. Işın izleme, nesnelerin şekillerine dayalı olarak çarpışmaları tespit etmek için yaygın bir yöntemdir. Ancak, ışın izlemenin GPU üzerinde hızlı çalışacak şekilde uygulanması zordur. Işın izlemenin basit versiyonları yavaştır ve daha çok CPU simülasyonları için uygundur.

Başka bir yöntem ise dünyayı ayrıklaştırmak ve yalnızca yakın nesnelerle çarpışmaları kontrol etmektir. Yukarıda tanımlanan dinamik modele dayanarak, nesneler arasındaki mesafeyi hesaplamak yeterlidir.

CAMAR'da, her nesne bir çember olarak temsil edilir. Bu tercih birkaç avantaj sağlar. İki çember arasındaki mesafeyi kontrol etmek basit ve hızlıdır. Işın izlemenin gerektirdiği özel durumlara ihtiyaç duymaz. Ayrıca, dikdörtgenler veya çokgenler gibi farklı şekillerin işlenmesinin karmaşıklığını önler. Bundan dolayı, simülasyon birçok ajanla aynı anda kolayca GPU'lar üzerinde çalıştırılabilir. Bu tasarım, CAMAR'ın büyük ölçekli çok ajanlı görevleri verimli ve yüksek performansla simüle etmesini sağlar.

Harita Üreteçleri CAMAR'daki her nesne bir çember olarak temsil edilse de, karmaşık ve detaylı haritalar oluşturmak hâlâ mümkündür. Birçok küçük çemberin birbirine yakın şekilde birleştirilmesiyle karmaşık yapılar oluşturulabilir. Kullanılan çember sayısı arttıkça harita daha doğru hale gelir. Bu yöntem, temel eleman her zaman çember olmasına rağmen, kullanıcıların duvarlar, tüneller, labirentler ve diğer karmaşık şekilleri simüle etmesine olanak tanır.

CAMAR, birkaç farklı türde yerleşik harita içerir. Ayrıca, JAX zamanında derleme (just-in-time compilation) [30] ile uyumlu olmasalar dahi kullanıcıların özel harita üreteçleri eklemesine imkân sağlar. Özel harita üreteci, string grid ya da toplu dizge ızgarası formatlarını kullanarak kolayca bağlanabilir. Bu esneklik, kullanıcıların basit rastgele düzenlemelerden karmaşık ve gerçekçi haritalara kadar çeşitli ortamlar tasarlamasına olanak tanır.

Mevcut yerleşik haritalar ve üreteç seti şunları içerir (bkz. Şekil 4):

4a rastgele ızgara: Engellerin, ajanların ve hedeflerin önceden belirlenmiş boyutta bir ızgara üzerinde rastgele yerleştirildiği bir harita.

4b labmaze ızgarası: LabMaze kullanılarak oluşturulan haritalar[5] [31] - bağlantılı odalara sahip bir labirent oluşturucu.

4c `movingai`: MovingAI karşılaştırma ölçütünden entegre edilmiş iki boyutlu haritalar [29], sürekli planlama görevlerine uyarlanmıştır. Ayrıca toplu dizge ızgarası biçiminde de kullanılabilirler.

4d mağaralar devamı: Perlin gürültüsü kullanılarak oluşturulan mağaraların bulunduğu sürekli tipte bir harita; video oyunlarında gerçekçi ve çeşitli manzaralar yaratmak için yaygın bir yöntemdir.

4e dizge ızgarası: Metin düzenine dayanan bir ızgara haritası. Engeller, MovingAI karşılaştırma ölçütüne [29] benzer şekilde bir dizgedeki karakterlere göre yerleştirilir. Ajan ve hedef pozisyonları, dizgedeki boş hücrelere bağlı olarak sabit veya rastgele olabilir.

4f toplu dizge ızgarası: dizge ızgarasına benzer, ancak paralel ortamlarda farklı engel düzenlerini destekler. Bu, aynı anda birden çok harita varyasyonu üzerinde eğitim yapılmasına olanak sağlar.

Ödül Fonksiyonu CAMAR, her ajan için her zaman adımında skalar bir ödül kullanmaktadır. Bu ödül dört bileşenin toplamıdır: bir hedef ödülü, bir çarpışma cezası, hareket-temelli bir ödül ve toplu başarı ödülü:

[5] https://github.com/google-deepmind/labmaze

$$r_i(t) = r_{\text{all\_g}}(t) + r_{\text{on\_g}_i}(t) + r_{\text{collision}_i}(t) + r_{\text{g\_dist}_i}(t)$$

The terms are defined as follows:

$$\begin{cases} r_{\text{all\_g}}(t) = +0.5, \text{ if } \forall i : \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\| \leq R_{\text{g}}; \\ r_{\text{on\_g}_i}(t) = +0.5, \text{ if } \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\| \leq R_{\text{g}}; \\ r_{\text{collision}_i}(t) = -1, \text{ if } \exists j : \|\Delta\vec{x}_{ij}(t)\| < d_{\min}; \\ r_{\text{g\_dist}_i}(t) = +\texttt{shaping} \cdot \\ \quad \cdot \left(\|\vec{x}_i(t-dt) - \vec{x}_{\text{g}_i}\| - \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\|\right) \end{cases}$$

Here, $\vec{x}_i(t)$ is the position of agent i at time t, $\vec{x}_{\text{g}_i}$ is the goal position for agent i, R_{g} is the distance threshold to count as reaching the goal (goal radius), $\Delta\vec{x}_{ij}(t)$ is the vector between agent i and another object j, $d_{\min}$ is the minimal distance between agent i and object j (for circle objects $d_{\min} = R_i + R_j$ where R_i and R_j are radii), `shaping` is a user-defined coefficient that controls the strength of the movement-based term.

To support cooperation, the environment gives an extra reward when all agents reach their goals. In this case, each agent receives an additional reward of +0.5.

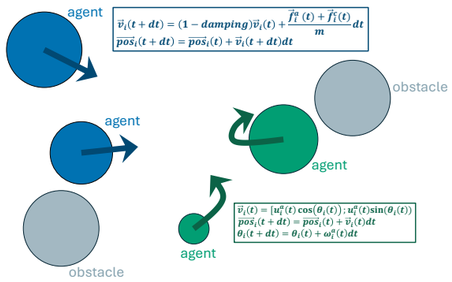


Figure 5: Illustration of heterogeneous agents with different sizes and dynamics supported by CAMAR. Blue agents are governed by `HolonomicDynamic`, while green agents follow `DiffDriveDynamic`. All agents navigate a shared environment while avoiding gray obstacles.

Heterogeneous agents Additionally, CAMAR supports heterogeneous agents in both size and dynamics. All map generators can produce agents with different properties, making it possible to study diverse multi-agent systems inspired by real-world scenarios.

For example, some agents can use `HolonomicDynamic`, while others follow `DiffDriveDynamic`. These agents operate together in a shared space, interact with the same obstacles, and must coordinate their movements despite having different control rules (Fig. 5). Each agent follows its own dynamics model, but all agents contribute to a single global simulation.

CAMAR also supports agents with different sizes. Each agent can have its own radius, which affects how it moves and avoids collisions. This adds complexity to coordination and planning.

Metrics In multi-agent pathfinding research, it is common to use metrics such as success rate (SR), flowtime (FT), makespan (MS), and coordination (CO) to compare different methods. We adopt these metrics in our work and define them more formally as follows in the equation below:

$$\begin{cases} \text{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\|\vec{x}_i(T) - \vec{x}_{\text{goal}}\| \leq R_{\text{g}}\}; \\ \text{FT} = \frac{1}{N}\sum_{i=1}^{N} t_i; \\ \text{MS} = \max_{i=1,\ldots,N} t_i; \\ \text{CO} = 1 - \frac{C}{N \times T} \end{cases}$$

Here, N is the number of agents, T is the maximum episode length, t_i is the time step when agent i first reaches its goal, C is the total number of collisions over the episode.

Evaluation Protocols To support rigorous and reproducible benchmarking, CAMAR includes a standardized suite of evaluation protocols inspired by and extending prior work on cooperative MARL evaluation [32]. We adapt that framework for continuous multi-agent pathfinding, focusing on generalization across both agent count and map structure.

We propose three evaluation tiers: **Easy**, **Medium**, and **Hard**, each targeting a different level of generalization. **Easy** evaluates performance on unseen start and goal positions using the same map type and number of agents as during training. **Medium** tests generalization to maps with similar structure but a different number of agents and obstacle parameters. **Hard** measures generalization to fully unseen map types from the MovingAI street collection, often with a different number of agents than during training.

Each tier follows a defined training and evaluation setup using introduced metrics, aggregated by the Interquartile Mean (IQM) with 95% confidence intervals (CI_{95}) for fair comparison across methods and difficulty levels.

All experiments use fixed JAX [30] random seeds for reproducibility. In total, each protocol involves training multiple models and running thousands of evaluation episodes. We include detailed training and evaluation scripts and provide all protocol maps in the public CAMAR repository.

These protocols help the community better track progress on continuous multi-agent pathfinding and identify which algorithms generalize well to more realistic conditions. Sample efficiency curves, metric-vs-agent-count plots, and performance profiles are supported and recommended for deeper analysis.

Experimental Evaluation

In this section, we evaluate both the scalability and benchmarking capabilities of our environment. We begin by training and testing a set of popular MARL algorithms, as well as classical non-learnable and hybrid methods. These experiments show that the environment supports a wide range of navigation and coordination strategies. We also present results from a simple heterogeneous-agent scenario to demonstrate support for heterogeneous MARL research. Finally, we

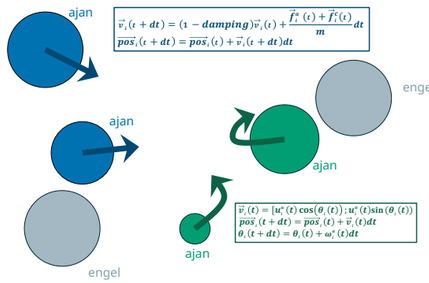
$$r_i(t) = r_{\text{all\_g}}(t) + r_{\text{on\_g}_i}(t) + r_{\text{collision}_i}(t) + r_{\text{g\_dist}_i}(t)$$

Terimler aşağıdaki biçimde tanımlanmıştır:

$$\begin{cases} r_{\text{all\_g}}(t) = +0.5, \text{ eğer } \forall i \; \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\| \leq R_{\text{g}}; \\ r_{\text{on\_g}_i}(t) = +0.5, \text{ eğer } \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\| \leq R_{\text{g}}; \\ r_{\text{çarpışma}_i}(t) = -1, \text{ if } \exists j : \|\Delta\vec{x}_{ij}(t)\| < d_{\min}; \\ r_{\text{g\_mesafe}_i}(t) = +\text{şekillendirme} \cdot \\ \quad \cdot \left(\|\vec{x}_i(t-dt) - \vec{x}_{\text{g}_i}\| - \|\vec{x}_i(t) - \vec{x}_{\text{g}_i}\|\right) \end{cases}$$

Burada, $\vec{x}_i(t)$, ajanının konumudur i zamanında t, $\vec{x}_{\text{g}_i}$ sudur i nci ajan için hedef pozisyonu i, R_{g} mesafe eşik değeri hedefe ulaşma olarak saymak için mesafe eşiğidir (hedef yarıçapı). $\Delta\vec{x}_{ij}(t)$ ajan i ile başka bir nesne j $d_{\min}$ ajan ile nesne arasında minimal mesafedir i ve nesne j (çember nesneler için $d_{\min} = R_i + R_j$ burada $R_i$ve R_j yarıçaplardır), şekillendirme hareket bazlı terimin gücünü kontrol eden kullanıcı tanımlı bir katsayıdır.

İşbirliğini desteklemek için ortam, tüm ajanlar hedeflerine ulaştığında ekstra ödül verir. Bu durumda, her ajan ek olarak +0,5 ödül alır.



Şekil 5: CAMAR tarafından desteklenen, farklı boyut ve dinamiklere sahip heterojen ajanların gösterimi. Mavi ajanlar HolonomikDinamik tarafından yönetilirken, yeşil ajanlar DiffDriveDynamic'i takip etmektedir. Tüm ajanlar ortak bir ortamda hareket ederken gri engellerden kaçınır.

Heterojen ajanlar Bunun yanı sıra, CAMAR hem boyut hem de dinamik açısından heterojen ajanları desteklemektedir. Tüm harita oluşturucular, farklı özelliklere sahip ajanlar üretebilmekte ve böylece gerçek dünya senaryolarından esinlenen çeşitli çok ajanlı sistemlerin incelenmesini mümkün kılmaktadır.

Örneğin, bazı ajanlar HolonomikDinamik kullanabilirken, diğerleri DiffDriveDynamic'i takip edebilir. Bu ajanlar ortak bir alanda birlikte çalışır, aynı engellerle etkileşime girer ve farklı kontrol kurallarına sahip olmalarına rağmen hareketlerini koordine etmek zorundadırlar (Şekil 5). Her ajan kendi dinamik modelini takip eder, ancak tüm ajanlar tek bir küresel simülasyona katkı sağlamaktadır.

CAMAR ayrıca farklı boyutlardaki ajanları da desteklemektedir. Her ajan, hareket etme biçimini ve çarpışmalardan kaçınmayı etkileyen kendine özgü bir yarıçapa sahip olabilir. Bu durum, koordinasyon ve planlamaya ek bir karmaşıklık kazandırmaktadır.

Metrikler Çok etmenli yol bulma araştırmalarında, farklı yöntemleri karşılaştırmak amacıyla başarı oranı (SR), akış zamanı (FT), tamamlanma süresi (MS) ve koordinasyon (KO) gibi metrikler yaygın olarak kullanılmaktadır. Çalışmamızda bu metrikleri benimseyerek, aşağıdaki denklemde daha resmi bir şekilde tanımlamaktayız:

$$\begin{cases} \text{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\|\vec{x}_i(T) - \vec{x}_{\text{goal}}\| \leq R_{\text{g}}\}; \\ \text{FT} = \frac{1}{N}\sum_{i=1}^{N} t_i; \\ \text{MS} = \max_{i=1,\ldots,N} t_i; \\ \text{KO} = 1 - \frac{C}{N \times T} \end{cases}$$

Burada, N ajan sayısını, T maksimum bölüm uzunluğunu, t_i ajan i 'nin hedefe ilk ulaştığı zaman adımını, C bölüm boyunca meydana gelen toplam çarpışma sayısını ifade etmektedir.

Değerlendirme Protokolleri Yayınlanabilir ve yeniden üretilebilir karşılaştırmalar yapmak amacıyla, CAMAR kooperatif Çok Etmenli Takviyeli Öğrenme (MARL) değerlendirmesine ilişkin önceki çalışmaları temel alarak ve geniş leterek standartlaştırılmış bir değerlendirme protokolleri paketi sunmaktadır [32]. Sürekli çok etmenli yol bulma için, hem ajan sayısı hem de harita yapısı genelinde genelleme üzerine odaklanarak bu çerçeveyi uyarlıyoruz.

Üç değerlendirme seviyesi öneriyoruz: Kolay, Orta ve Zor; her biri farklı bir genelleme düzeyini hedeflemektedir. Kolay, eğitim sırasında kullanılan ile aynı harita türü ve ajan sayısı kullanılarak görülmemiş başlangıç ve hedef pozisyonlarında performansı değerlendirir. Orta, benzer yapıya sahip fakat farklı ajan sayısı ve engel parametrelerine sahip haritalarda genelleme becerisini test eder. parametrelerine. Zor, MovingAI sokak koleksiyonundan tamamen yeni harita türlerine genellemeyi ölçer; bu haritalarda ajan sayısı genellikle eğitimdekinden farklıdır.

Her seviye, yöntemler ve zorluk seviyeleri arasında adil karşılaştırma sağlamak için %95 güven aralığı (CI_{95}) ile Interquartile Mean (IQM) yöntemi kullanılarak tanımlanmış eğitim ve değerlendirme düzeneklerini takip eder.

Tüm deneylerde tekrarlanabilirlik için sabit JAX [30] rastgele tohumları kullanılmıştır. Toplamda, her protokol çok sayıda modeli eğitmeyi ve binlerce değerlendirme bölümü çalıştırmayı içermektedir . Detaylı eğitim ve değerlendirme betikleri dahil edilmiş ve tüm protokol haritaları kamuya açık CAMAR deposunda sunulmaktadır.

Bu protokoller, topluluğun sürekli çok etmenli yol bulma alanındaki ilerlemeyi daha iyi takip etmesine ve hangi algoritmaların daha gerçekçi koşullara iyi genelleştiğini belirlemesine olanak sağlamaktadır. Örnek verimlilik eğrileri, metrik-ajansayısı grafikler ve performans profilleri desteklenmekte ve daha derin analizler için önerilmektedir.

Deneysel Değerlendirme

Bu bölümde, ortamımızın ölçeklenebilirliğini ve karşılaştırma ölçütü yeteneklerini değerlendirmekteyiz. Popüler MARL algoritmalarının yanı sıra klasik öğrenmeyen ve karma yöntemleri eğitip test ederek başlamaktayız. Bu deneyler, ortamın geniş bir navigasyon ve koordinasyon stratejileri yelpazesini desteklediğini ortaya koymaktadır. Ayrıca, heterojen MARL araştırmalarını desteklemek amacıyla basit bir heterojen ajan senaryosundan elde edilen sonuçları sunuyoruz. Son olarak,

Algorithm	random_grid				labmaze_grid			
	SR ↑	FT ↓	MS ↓	CO ↑	SR ↑	FT ↓	MS ↓	CO ↑
IPPO	0.410±0.001	1695±10	160.0±0.0	1.000±0.000	0.213±0.013	2104±14	160.0±0.0	1.000±0.000
MAPPO	0.830±0.001	984±5	151.4±0.3	1.000±0.000	0.568±0.004	1484±8	160.0±0.0	1.000±0.000
IDDPG	0.335±0.001	1851±10	160.0±0.0	1.000±0.000	0.167±0.000	2772±14	160.0±0.0	0.996±0.000
MADDPG	0.041±0.000	2508±12	160.0±0.0	0.913±0.001	0.027±0.000	2745±12	160.0±0.0	0.854±0.001
ISAC	0.115±0.001	2523±14	160.0±0.0	1.000±0.000	0.047±0.000	2808±12	160.0±0.0	1.000±0.000
MASAC	0.281±0.001	1843±11	160.0±0.0	0.856±0.001	0.105±0.001	2098±12	160.0±0.0	0.781±0.001
RRT*+IPPO	0.420±0.001	1426±9	160.0±0.0	1.000±0.000	0.511±0.001	1316±6	160.0±0.0	0.999±0.000
RRT*+MAPPO	0.828±0.001	971±5	150.4±0.3	1.000±0.000	0.556±0.001	1326±7	160.0±0.0	0.999±0.000
RRT*+IDDPG	0.280±0.001	2181±12	160.0±0.0	1.000±0.000	0.189±0.000	2635±14	160.0±0.0	0.997±0.000
RRT*+MADDPG	0.037±0.000	2953±15	160.1±0.0	0.984±0.000	0.037±0.000	2918±14	160.1±0.0	0.969±0.000
RRT*+ISAC	0.143±0.000	2618±13	160.0±0.0	1.000±0.000	0.058±0.000	2749±13	160.0±0.0	1.000±0.000
RRT*+MASAC	0.054±0.000	2511±14	160.0±0.0	1.000±0.000	0.034±0.000	2854±15	160.0±0.0	0.994±0.000
RRT*+PD	0.678±0.002	2010±59	160.0±0.0	0.997±0.000	0.692±0.004	1807±49	160.0±0.0	0.971±0.002
RRT+PD	0.413±0.014	2440±264	160.0±0.0	0.788±0.021	0.528±0.021	2049±251	160.0±0.0	0.558±0.025

Table 2: Extended performance comparison across different algorithms in the `random_grid` and `labmaze_grid` environments including integrated RRT* with off-policy algorithms additionally. Reported metrics are SR (Success Rate), FT (Flowtime), MS (Makespan), and CO (Coordination), each shown as IQM±CI$_{95}$. Confidence intervals are symmetric for clarity and computed using 1K bootstrapped samples. Arrows indicate the direction of improvement: ↑ denotes higher is better, ↓ indicates lower is better. Tan boxes highlight the best-performing approach. The CO metric is not colored here for visibility.

measure the performance of the simulator in terms of simulation speed, and compare it with VMAS using a shared experimental setup.

Experimental Setup

We evaluate six MARL algorithms: IPPO, MAPPO, IDDPG, MADDPG, ISAC [33], and MASAC. In addition, we include two non-learnable baselines, RRT+PD and RRT*+PD, and two hybrid methods, RRT*+IPPO and RRT*+MAPPO.

All methods are evaluated on two procedurally generated map types: `random_grid` and `labmaze_grid`, each with 6 versions varying in obstacle density and agent count (8 or 32). For `labmaze_grid`, an additional connection probability from 0.4 to 1.0 tests different maze complexities. Generation details for training and evaluation maps are available in the Appendix D.

The "independent" variants (IPPO [1], IDDPG, ISAC) train each agent using its own policy and value function. These methods do not use centralized critics or information sharing across agents. In contrast, the multi-agent versions (MAPPO, MADDPG, MASAC) use centralized critics during training to improve coordination. All approaches use parameter sharing, meaning that agents use the same neural network weights.

Each algorithm is trained for 20M (IPPO, MAPPO) and 2M (IDDPG, MADDPG, ISAC, MASAC) steps per scenario. The training is done independently for each of the 12 map variations. After training, we evaluate the models on both seen and unseen tasks to test their generalization. In total, we train 532 models and evaluate them across 5184 tasks, with 1000 episodes per task. The experiments were run on a single NVIDIA H100 GPU and took around 1000 hours in total.

For the non-learning baselines, we use RRT+PD and RRT*+PD. These methods use classical planning algorithms to generate a path to the goal for each agent. Each path is generated using either RRT (with 50,000 iterations) [34] or RRT* (with 3000 iterations). The agent then follows the path using a simple PD controller.

We also evaluate hybrid methods where agents receive additional RRT* information during training and evaluation. At the start of each episode, RRT* generates sample paths from the goal to the agent's position. These paths and their estimated costs are included in the agent's observation, enabling the policy to learn from approximate cost-to-go values without invoking RRT* at every step.

To evaluate simulator performance, we measure simulation speed in steps per second (SPS) on a 20×20 `random_grid` map with 0.3 obstacle density (120 obstacles). We vary the number of agents and parallel environments to assess CA-MAR's scalability. For fair comparison, we benchmark CA-MAR against VMAS [14] using identical map size, agent count, and a single NVIDIA H100 GPU.

Benchmark

The main results are presented in Table 2, which reports success rate (SR), flowtime (FT), makespan (MS), and coordination (CO) for each algorithm on two types of maps.

On the `random_grid` map, MAPPO achieves the highest SR, with good FT and full CO. Its centralized critic helps agents plan jointly and share useful information. RRT*+MAPPO shows similar SR but performs better in FT and MS, which shows that adding planning support improves movement efficiency.

RRT*+IPPO also improves over IPPO alone, although the gains are smaller. RRT*+PD achieves high SR without learning, but it performs worse in CO. This is expected because it plans for each agent independently. The planner uses full map knowledge but does not consider other agents during

Algoritma	rastgele ızgara				labmaze ızgarası			
	SR ↑	FT ↓	MS ↓	KO ↑	SR ↑	FT ↓	MS ↓	KO ↑
IPPO	0.410±0.001	1695±10	160.0±0.0	1.000±0.000	0.213±0.013	2104±14	160.0±0.0	1.000±0.000
MAPPO	0.830±0.001	984±5	151.4±0.3	1.000±0.000	0.568±0.004	1484±8	160.0±0.0	1.000±0.000
IDDPG	0.335±0.001	1851±10	160.0±0.0	1.000±0.000	0.167±0.000	2772±14	160.0±0.0	0.996±0.000
MADDPG	0.041±0.000	2508±12	160.0±0.0	0.913±0.001	0.027±0.000	2745±12	160.0±0.0	0.854±0.001
ISAC	0.115±0.001	2523±14	160.0±0.0	1.000±0.000	0.047±0.000	2808±12	160.0±0.0	1.000±0.000
MASAC	0.281±0.001	1843±11	160.0±0.0	0.856±0.001	0.105±0.001	2098±12	160.0±0.0	0.781±0.001
RRT*+IPPO	0.420±0.001	1426±9	160.0±0.0	1.000±0.000	0.511±0.001	1316±6	160.0±0.0	0.999±0.000
RRT*+MAPPO	0.828±0.001	971±5	150.4±0.3	1.000±0.000	0.556±0.001	1326±7	160.0±0.0	0.999±0.000
RRT*+IDDPG	0.280±0.001	2181±12	160.0±0.0	1.000±0.000	0.189±0.000	2635±14	160.0±0.0	0.997±0.000
RRT*+MADDPG	0.037±0.000	2953±15	160.1±0.0	0.984±0.000	0.037±0.000	2918±14	160.1±0.0	0.969±0.000
RRT*+ISAC	0.143±0.000	2618±13	160.0±0.0	1.000±0.000	0.058±0.000	2749±13	160.0±0.0	1.000±0.000
RRT*+MASAC	0.054±0.000	2511±14	160.0±0.0	1.000±0.000	0.034±0.000	2854±15	160.0±0.0	0.994±0.000
RRT*+PD	0.678±0.002	2010±59	160.0±0.0	0.997±0.000	0.692±0.004	1807±49	160.0±0.0	0.971±0.002
RRT+PD	0.413±0.014	2440±264	160.0±0.0	0.788±0.021	0.528±0.021	2049±251	160.0±0.0	0.558±0.025

Tablo 2: Entegre RRT* ve off-policy algoritmaların da dahil olduğu, rastgele ızgara ve labmaze ızgarası ortamlarındaki farklı algoritmaların kapsamlı performans karşılaştırması. Rapor edilen metrikler SR (Başarı Oranı), FT (Akış Süresi), MS (Bitiş Süresi) ve KO (Koordinasyon) olup, her biri IQM±CI 95 olarak sunulmuştur. Güven aralıkları açıklık için simetrik olup 1.000 bootstrap örneği kullanılarak hesaplanmıştır. Oklar iyileşme yönünü gösterir: ↑ daha yüksek değer daha iyidir, ↓ daha düşük değer daha iyidir. Bej kutular en iyi performans gösteren yaklaşımı vurgulamaktadır. Görünürlüğü artırmak amacıyla KO metriği burada renklendirilmemiştir.

Simülasyon hızı açısından simülatörün performansını ölçerek, paylaşılan deneysel kurulum kullanılarak VMAS ile karşılaştırılmasını gerçekleştirdik.

Deneysel Kurulum

Altı MARL algoritmasını değerlendirdik: IPPO, MAPPO, IDDPG, MADDPG, ISAC [33] ve MASAC. Ayrıca, iki öğrenilemeyen temel yaklaşım olan RRT+PD ve RRT*+PD ile iki karma yöntemi, RRT* +IPPO ve RRT*+MAPPO, dâhil ettik.

Tüm yöntemler, engel yoğunluğu ve ajan sayısının (8 veya 32) değiştiği altı versiyonu bulunan iki prosedürel olarak oluşturulmuş harita türü üzerinde değerlendirilmiştir: `random grid` ve `labmaze grid`. `Labmaze grid` için, 0.4 ile 1.0 arasında değişen ek bir bağlantı olasılığı farklı labirent karmaşıklıklarını test etmek amacıyla kullanılmıştır. Eğitim ve değerlendirme haritalarının oluşturulma detayları Ek D'de sunulmuştur.

"Bağımsız" varyantlar (IPPO [1], IDDPG, ISAC), her ajanın kendi politika ve değer fonksiyonunu kullanarak eğitim yapmaktadır. Bu yöntemler merkezi eleştirmenleri veya ajanlar arasındaki bilgi paylaşımını kullanmamaktadır. Buna karşılık, çok ajanlı varyantlar (MAPPO, MADDPG, MASAC), koordinasyonu artırmak amacıyla eğitim sırasında merkezi eleştirmenleri kullanmaktadır. Tüm yaklaşımlar parametre paylaşımını kullanmakta olup, ajanlar aynı sinir ağı ağırlıklarını kullanmaktadır.

Her algoritma, senaryo başına sırasıyla 20M (IPPO, MAPPO) ve 2M (IDDPG, MADDPG, ISAC, MASAC) adım boyunca eğitilmektedir. Eğitim, 12 farklı harita varyasyonu için bağımsız olarak gerçekleştirilmiştir. Eğitim sonrasında, modeller hem daha önce görülmüş hem de görülmemiş görevlerde genelleme kabiliyetlerini test etmek üzere değerlendirilmiştir. Toplamda, 532 model eğitilmiş ve 5184 görevde, görev başına 1000 bölüm olmak üzere değerlendirilmiştir. Deneyler, tek bir NVIDIA H100 GPU üzerinde yürütülmüş ve toplamda yaklaşık 1000 saat sürmüştür.

Eğitim yapmayan temel yöntemler için RRT+PD ve RRT*+PD kullanıyoruz. Bu yöntemler klasik planlama algoritmalarını kullanır her ajan için hedefe giden bir yol oluşturmak amacıyla. Her yol, RRT (50.000 iterasyon ile) [34] veya RRT* (3000 iterasyon ile) kullanılarak oluşturulur. Ajan daha sonra yolu basit bir PD kontrolörü ile izler.

Ayrıca ajanların eğitim ve değerlendirme sırasında ek RRT* bilgisi aldığı karma yöntemleri de değerlendiriyoruz.

Her bölümün başında, RRT* hedeften ajanın konumuna örnek yollar oluşturur. Bu yollar ve tahmini maliyetleri, ajanın gözlemlerine dahil edilir; böylece politika, her adımda RRT* çağrısı yapmadan yaklaşık maliyet-ilerleme değerlerinden öğrenebilir.

Simülatör performansını değerlendirmek için 0.3 engel yoğunluğuna (120 engel) sahip 20×20 rastgele ızgara haritasında adım/saniye (SPS) cinsinden simülasyon hızını ölçüyoruz. CA-MAR'ın ölçeklenebilirliğini değerlendirmek için ajan sayısı ve paralel ortam sayısını değiştiriyoruz. Adil karşılaştırma için, CA-MAR 'ı VMAS [14] ile aynı harita boyutu, ajan sayısı ve tek bir NVIDIA H100 GPU kullanılarak karşılaştırma ölçütünde değerlendirdik

Karşılaştırma Ölçütü

Ana sonuçlar, her algoritmanın iki harita türündeki başarı oranı (SR), akış süresi (FT), tamamlanma süresi (MS) ve koordinasyon (KO) değerlerini içeren Tablo 2'de sunulmaktadır.

Rastgele ızgara haritasında MAPPO, en yüksek SR'yi elde etmiş, iyi bir FT ve tam CO sağlamıştır. Merkezi eleştirmeni, ajanların ortak planlama yapmasına ve faydalı bilgileri paylaşmasına olanak tanır. RRT*+MAPPO benzer SR değerleri gösterirken, FT ve MS açısından daha iyi performans sergilemekte; bu da planlama desteğinin hareket verimliliğini artırdığını göstermektedir.

RRT*+IPPO, tek başına IPPO'ya kıyasla gelişim göstermekte ancak kazançlar daha sınırlıdır. RRT*+PD öğrenme gerektirmeden yüksek SR sağlamaktadır ancak KO açısından daha düşük performans sergilemektedir; bu, her ajan için bağımsız planlama yapılmasından kaynaklanmaktadır. Planlayıcı tam harita bilgisini kullanmakla birlikte diğer ajanları dikkate almamaktadır.

execution. As a result, agents sometimes collide in narrow areas.

RRT* with Off-Policy Algorithms We also evaluate hybrid methods that combine RRT* with off-policy MARL algorithms. As shown in Table 2, the results are mixed. Some methods benefit from additional planning information. For example, RRT*+ISAC slightly improves over ISAC alone in both success rate (SR) and coordination (CO), especially in the `random_grid` setting. RRT*+IDDPG also shows moderate improvement, achieving better SR and lower flowtime (FT) than IDDPG.

However, in other cases, the integration leads to weaker results. RRT*+MASAC and RRT*+MADDPG perform poorly, with SR staying below 0.05 in both maps. One possible reason is the increased input size to the models. Off-policy algorithms with centralized critics, such as MASAC and MADDPG, must process long input vectors that include observations and RRT* features from all agents. This can make training unstable and reduce generalization.

Overall, while combining RRT* with off-policy methods can help in some settings, the results suggest that better integration strategies or more expressive policy architectures may be needed to fully benefit from hybrid approaches.

Among MARL methods, IPPO and IDDPG perform well in SR, but their FT is worse than MAPPO. ISAC shows high CO but low SR, possibly due to its focus on smooth motion and risk reduction. MASAC and MADDPG perform poorly. In particular, MADDPG shows very low SR and fails to explore even in the simpler tasks. We investigate these issues in more detail in the Appendix D.

The `labmaze_grid` map is more challenging. Narrow passages make reward signals sparse. In this setting, SR drops across all MARL methods. RRT*+PD performs the best in SR, showing the value of full-path planning when learning signals are limited. However, its CO score remains lower than MARL baselines.

The simpler RRT+PD baseline performs worse than RRT*+PD across all metrics. It produces longer and less efficient paths because RRT does not optimize for path cost and often generates jagged routes. On the `labmaze_grid`, RRT+PD still performs better in SR than many learning-based methods, but with higher FT and lower CO. This result shows that even basic planning methods can outperform MARL baselines in sparse reward environments, but their inability to consider other agents during execution limits coordination and overall efficiency.

These findings highlight a trade-off. Planning methods like RRT*+PD solve complex tasks more reliably, while learning-based policies handle coordination better. To explore this further, we evaluated hybrid methods. These approaches combine RRT* samples with MARL observations to guide exploration. However, results show only small gains or similar performance. One possible reason is the size of the observation vector, which may be too large for the simple policy network to process effectively. Future work may improve integration with better architectures. See Appendix D for more detailed analysis of performance of algorithms.

Heterogeneous Agents

To show CAMAR's support for heterogeneous agents, we modify a simple coordination task called `give_way`. In this new version, two agents must pass through a narrow corridor, but only the smaller red agent can fit into the central chamber. The larger blue agent cannot enter this area and must wait for the other to pass.

We evaluate both standard and heterogeneous versions of IPPO and MAPPO. In the heterogeneous versions (HetIPPO and HetMAPPO), agents use separate policy models. Results in Fig. 6b show that HetIPPO performs better than its shared-policy version. However, HetMAPPO fails, likely because the centralized critic struggles with large and diverse input spaces.

(a) `hetero_give_way` scenario

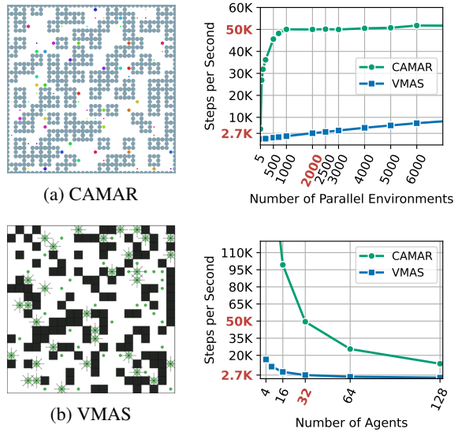
Algorithm	SR
IPPO	0.5
HetIPPO	0.7
MAPPO	0.5
HetMAPPO	0.0

(b) SR results.

Figure 6: Example of heterogeneous agent coordination (a). Success rates of algorithms are shown in (b).

This experiment shows that CAMAR supports agents with different sizes and models. It can be used to study coordination and policy learning in heterogeneous teams. More details and discussion are provided in the Appendix D.

Scalability Analysis



(a) CAMAR

(b) VMAS

Figure 7: Scalability comparison between CAMAR and VMAS across different settings, evaluating the impact of increasing parallel environments, agent count, and obstacles on simulation speed.

yürütme. Sonuç olarak, ajanlar bazen dar alanlarda çarpışmaktadır.

Off-Policy Algoritmalar ile RRT* Ayrıca, RRT* ile off-policy MARL algoritmalarını birleştiren karma yöntemleri değerlendirmekteyiz. Tablo 2'de gösterildiği gibi, sonuçlar karışıktır. Bazı yöntemler ek planlama bilgisinden fayda sağlamaktadır. Örneğin, RRT*+ISAC özellikle rastgele ızgara ortamında, hem başarı oranında (SR) hem de koordinasyonda (KO), yalnızca ISAC'a kıyasla hafif bir iyileşme göstermektedir. RRT*+IDDPG de orta düzeyde bir iyileşme sergilemekte, IDDPG'den daha iyi SR ve daha düşük akış süresi (FT) elde etmektedir.

Ancak diğer durumlarda entegrasyon daha zayıf sonuçlara yol açmaktadır. RRT*+MASAC ve RRT*+MADDPG kötü performans göstermekte olup, her iki haritada da SR 0,05'in altında kalmaktadır . Bunun olası nedeni, modellere verilen girdi boyutunun artmasıdır . MASAC ve MADDPG gibi merkezi eleştirmenlere sahip off-policy algoritmalar, tüm ajanlardan gelen gözlemler ve RRT* özelliklerini içeren uzun girdi vektörlerini işlemelidir. Bu durum, eğitimi kararsız hale getirebilir ve genelleme yeteneğini düşürebilir.

Genel olarak, RRT* ile off-policy yöntemlerin birleştirilmesi bazı durumlarda fayda sağlamakla birlikte, sonuçlar karma yaklaşımlardan tam anlamıyla yararlanmak için daha iyi entegrasyon stratejilerine veya daha ifade edici politika mimarilerine ihtiyaç duyulduğunu göstermektedir.

MARL yöntemleri arasında IPPO ve IDDPG, SR performansında başarılı olurken FT performansları MAPPO'dan daha düşüktür. ISAC yüksek KO ancak düşük SR sergilemekte olup, bu durum muhtemelen hareketin akıcılığına ve riskin azaltılmasına odaklanmasından kaynaklanmaktadır. MASAC ve MADDPG ise zayıf performans göstermektedir.
Özellikle MADDPG çok düşük SR değerleri ortaya koymakta ve daha basit görevlerde dahi keşif yapamamaktadır. Bu sorunları Ek D'de daha ayrıntılı olarak incelemekteyiz.

Labmaze ızgarası haritası daha zorludur. Dar geçitler, ödül sinyallerinin seyrekleşmesine yol açmaktadır. Bu durumda, tüm MARL yöntemlerinde SR düşüşü gözlemlenmektedir. RRT*+PD, SR açısından en iyi performansı sergileyerek, öğrenme sinyallerinin sınırlı olduğu durumlarda tam yol planlamanın değerini ortaya koymaktadır. Ancak KO puanı , MARL temel yöntemlerinin altında kalmaya devam etmektedir.

Daha basit olan RRT+PD temel yöntemi, tüm metriklerde RRT*+PD'den daha kötü performans göstermektedir. RRT, yol maliyetini optimize etmediği ve sık sık keskin yollar oluşturduğu için daha uzun ve daha az verimli yollar üretmektedir. labmaze ızgarasında , RRT+PD hâlâ birçok öğrenme tabanlı yönteme kıyasla SR'de daha iyi performans gösterirken, daha yüksek FT ve daha düşük KO değerlerine sahiptir. Bu sonuç, temel planlama yöntemlerinin seyrek ödüllü ortamlarda MARL temel yöntemlerini geride bırakabileceğini, ancak yürütme sırasında diğer ajanları dikkate alamamalarının koordinasyon ve genel verimliliği sınırladığını ortaya koymaktadır.

Bu bulgular bir ödünleşmeyi ortaya koymaktadır. RRT*+PD gibi planlama yöntemleri karmaşık görevleri daha güvenilir çözerken, öğrenme tabanlı politikalar koordinasyonu daha etkin yönetmektedir. Bunu daha ayrıntılı incelemek için karma yöntemleri değerlendirdik. Bu yaklaşımlar keşfi yönlendirmek amacıyla RRT* örneklerini MARL gözlemleriyle birleştirmektedir. Ancak sonuçlar yalnızca sınırlı kazanımlar ya da benzer performans göstermektedir. Bunun olası nedeni , gözlem vektörünün boyutunun basit politika ağının etkin biçimde işlemesi için fazla büyük olması olabilir. Gelecek çalışmalar, daha gelişmiş mimarilerle entegrasyonun iyileştirilmesini amaçlamaktadır. Algoritmaların performansına ilişkin detaylı analiz için Ek D'ye bakınız.

Heterojen Ajanlar

CAMAR'ın heterojen ajanları desteklediğini göstermek amacıyla `give way` isimli basit bir koordinasyon görevini değiştirdik. Bu yeni versiyonda, iki ajan dar bir koridordan geçmelidir ancak sadece daha küçük olan kırmızı ajan merkezi bölmeye sığabilmektedir.
Daha büyük mavi ajan bu alana giremez ve diğer ajanın geçmesini beklemek zorundadır.

Hem standart hem de heterojen IPPO ve MAPPO versiyonlarını değerlendirdik. Heterojen versiyonlarda (HetIPPO ve HetMAPPO), ajanlar ayrı politika modelleri kullanmaktadır. Şekil 6b'deki sonuçlar , HetIPPO'nun ortak politika versiyonuna kıyasla daha iyi performans sergilediğini göstermektedir. Ancak HetMAPPO başarısız olmaktadır ; bu durum muhtemelen merkezi eleştirmenin büyük ve çeşitli giriş uzaylarında zorlanmasından kaynaklanmaktadır.

(a) `hetero give way` senaryosu

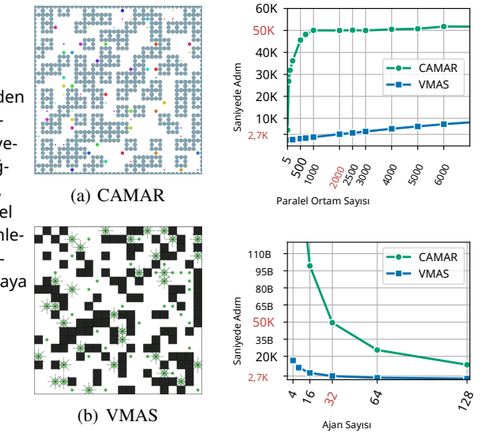
Algoritma	SR
IPPO	0.5
HetIPPO	0.7
MAPPO	0.5
HetMAPPO	0.0

(b) SR sonuçları.

Şekil 6: Heterojen ajan koordinasyon örneği (a). Algoritmaların başarı oranları (b) bölümünde gösterilmiştir.

Bu deney CAMAR'ın farklı boyut ve modellere sahip ajanları desteklediğini ortaya koymaktadır. Heterojen ekiplerde koordinasyon ve politika öğrenimini incelemek için kullanılabilir. Daha fazla detay ve tartışma Ek D'de sunulmaktadır.

Ölçeklenebilirlik Analizi



(a) CAMAR

(b) VMAS

Şekil 7: Paralel ortam sayısı, ajan sayısı ve engellerin artışının simülasyon hızı üzerindeki etkisinin değerlendirilmesi suretiyle CAMAR ve VMAS'ın farklı ayarlardaki ölçeklenebilirlik karşılaştırması.

We evaluate the scalability of our environment by measuring how the simulation speed changes when we increase the number of parallel environments, agents, and obstacles. Fig. 7 and 8 show the results of these experiments.

Fig. 7a shows a snapshot of the CAMAR environment with 32 agents on the 20×20 `random_grid` map. Agents move smoothly while avoiding the 120 fixed obstacles. Fig. 7b shows the same scenario in VMAS [14].

First, we measure the SPS as we increase the number of parallel environments from 5 to 7000, keeping 32 agents fixed. VMAS [14] scales almost linearly but reaches less than 10 000 SPS at 7000 environments. In contrast, CAMAR scales quickly up to 2000 environments and then maintains around 50 000 SPS even as more are added. This shows that CAMAR can simulate many parallel tasks without losing speed, which is important for fast training.

Next, we fix the number of environments at 2000 and vary the agent count from 4 to 128. CAMAR keeps over 100 000 SPS with fewer than 16 agents and remains above 10 000 SPS even with 128 agents. VMAS [14] starts near 20 000 SPS with 4 agents but drops sharply to 500 SPS at 128 agents. In this comparison, CAMAR is up to 20x faster than VMAS when many agents act at once.

Overall, CAMAR delivers much higher throughput and keeps strong performance as the number of agents increases. VMAS can boost its speed by adding more parallel environments, but it would need about 50 000 parallel environments to match the throughput that CAMAR provides at 2000 environments. We also compare CAMAR scalability with other modern MARL environments, which can be found in the Appendix E.

It also means that the total number of observations remains high, because each agent generates one observation per step.

Finally, we test how the number of obstacles affects scalability. We fix the number of agents at 32 and increase the number of obstacles up to 9920. The results, shown in Fig. 8, demonstrate that although SPS decreases with more obstacles, CAMAR still achieves about 15 400 SPS at the highest tested complexity. This indicates that CAMAR remains robust even in very cluttered environments.

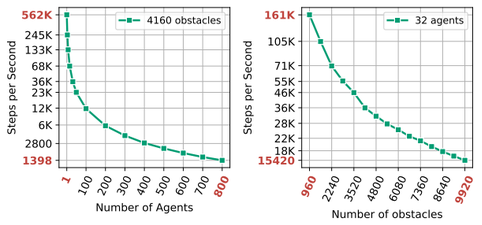


Figure 8: Scalability of CAMAR with increasing numbers of agents and obstacles under more extreme conditions. Despite a decrease in Steps per Second (SPS) as the number of agents grows to 800, CAMAR maintains approximately 1400 SPS, demonstrating its capability to simulate large multi-agent teams. Additionally, although SPS decreases, the total number of gathered observations remains high, as it scales proportionally with the number of agents.

We also study CAMAR under more extreme conditions. In Fig. 8, we increase the number of agents up to 800 while keeping the number of circle obstacles fixed at 4160. As the number of agents grows, the simulation speed drops, but our environment still maintains about 1400 SPS with 800 agents. This result shows that CAMAR can simulate very large multi-agent teams and still produce enough experience for learning.

Conclusion

This paper introduces CAMAR, a high-performance benchmark for continuous-space multi-agent reinforcement learning. CAMAR combines realistic dynamics with efficient simulation, supporting over 100 000 steps per second using JAX [30]. It includes a diverse set of navigation tasks, a standardized evaluation protocol with built-in metrics, and a range of strong baselines from both classical, learning-based, and hybrid methods. These components enable reliable, scalable, and reproducible evaluation of MARL algorithms.

References

[1] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[2] Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. In *AAMAS*, 2023.

[3] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

[4] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. Primal_2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

[5] Alexey Skrynnik, Anton Andreychuk, Maria Nesterova, Konstantin Yakovlev, and Aleksandr Panov. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17541–17549, 2024.

[6] Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. Mapf-gpt: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23126–23134, 2025.

[7] Brian Angulo, Aleksandr Panov, and Konstantin Yakovlev. Policy optimization to learn adaptive motion primitives in path planning with dynamic obstacles. *IEEE Robotics and Automation Letters*, 8(2):824–831, 2022.

Ortamımızın ölçeklenebilirliğini, paralel ortam sayısı, ajan sayısı ve engel sayısı arttıkça simülasyon hızının değişimini ölçerek değerlendiriyoruz.
Şekil 7 ve 8, bu deneylerin sonuçlarını göstermektedir.

Şekil 7a, 20 × 20 rastgele ızgara haritasında 32 ajanla CAMAR ortamının anlık görüntüsünü sunmaktadır. Ajanlar, 120 sabit engelden kaçınarak akıcı şekilde hareket etmektedir. Şekil 7b ise aynı senaryoyu VMAS [14] üzerinde göstermektedir.

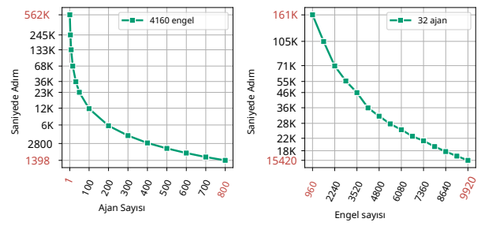
Öncelikle, ajan sayısı sabit tutulurken paralel ortam sayısı 5'ten 7000'e artırıldığında SPS değerini ölçüyoruz. VMAS [14] neredeyse doğrusal ölçeklenmekle birlikte, 7000 ortamda 10.000 SPS'nin altına düşmektedir. Buna karşılık, CAMAR hızla 2000 ortam sayısına ölçeklenmekte ve daha fazlası eklenmiş olsa bile yaklaşık 50.000 SPS değerini korumaktadır. Bu durum, CAMAR'ın birçok paralel görevi hız kaybetmeden simüle edebildiğini göstermektedir; bu da hızlı eğitim için önemlidir.

Sonraki aşamada ortam sayısını sabit tutarak 2000 , ajan sayısını 4 ile 128 arasında değiştirdik. CAMAR, 16 ajandan az olduğunda 100.000 SPS'nin üzerinde performans sergilerken, 128 ajanla bile 10.000 SPS'nin üzerinde kalmaktadır. VMAS [14], 4 ajanda yaklaşık olarak 20.000 SPS ile başlamakta ancak 128 ajan olduğunda hızla 500 SPS'ye düşmektedir. Bu karşılaştırmada, çok sayıda ajan aynı anda hareket ettiğinde CAMAR, VMAS'e kıyasla 20 kat daha hızlıdır.

Genel olarak CAMAR, çok daha yüksek verimlilik sunmakta ve ajan sayısı arttıkça performansını güçlü şekilde korumaktadır. VMAS, daha fazla paralel ortam ekleyerek hızını artırabilir ancak CAMAR'ın 2000 ortamda sağladığı verimliliğe ulaşmak için yaklaşık 50.000 paralel ortama ihtiyaç duyacaktır. Ayrıca CAMAR'ın ölçeklenebilirliğini diğer modern MARL ortamları ile karşılaştırıyoruz; bu karşılaştırma Ek E'de yer almaktadır.

Ayrıca, her ajan her adımda bir gözlem oluşturduğundan toplam gözlem sayısının yüksek kaldığı anlamına gelir.

Son olarak, engel sayısının ölçeklenebilirlik üzerindeki etkisini test ediyoruz. Ajan sayısını 32 olarak sabit tutup engel sayısını 9920 'ye kadar artırıyoruz. Şekil 8'de gösterilen sonuçlar, engel sayısı artsa da SPS'nin azaldığını, ancak CAMAR 'ın test edilen en yüksek karmaşıklıkta yaklaşık 15 400 SPS sağladığını göstermektedir. Bu durum, CAMAR'ın çok dağınık ortamlarda bile dayanıklı kaldığını göstermektedir.



Şekil 8: Daha aşırı koşullar altında ajan ve engel sayısının artmasıyla CAMAR'ın ölçeklenebilirliği. Ajan sayısı 800 'e çıktıkça Saniyedeki Adım Sayısı (SPS) azalsa da, CAMAR yaklaşık 1400 SPS değerini koruyarak büyük çok ajanlı ekipleri simüle edebilme kapasitesini göstermektedir. Ayrıca, SPS azalsa da, toplanan toplam gözlem sayısı yüksek kalmakta, çünkü bu sayı ajan sayısıyla orantılı olarak artmaktadır.

CAMAR'ı daha aşırı koşullar altında da incelemekteyiz.
Şekil 8'de, çember engel sayısını sabit tutarak ajan sayısını 800 'e çıkarmaktayız ve engel sayısı 4160 'dır. Ajan sayısı arttıkça simülasyon hızı düşmektedir, ancak ortamımız 800 ajanla bile yaklaşık 1400 SPS değerini korumaktadır.
Bu sonuç, CAMAR'ın çok büyük çok ajanlı ekipleri simüle edebildiğini ve öğrenme için yeterli deneyim üretebildiğini göstermektedir.

Sonuç

Bu makale, sürekli uzay çok ajanlı takviyeli öğrenme için yüksek performanslı bir karşılaştırma ölçütü olan CAMAR'ı tanıtmaktadır . CAMAR, gerçekçi dinamikleri verimli simülasyonla birleştirerek JAX [30] kullanarak saniyede 100 000 adımın üzerinde destek sağlar . Çeşitli navigasyon görevleri, yerleşik metriklere sahip standartlaştırılmış bir değerlendirme protokolü ve klasik, öğrenmeye dayalı ve karma yöntemlerden güçlü temel modeller içermektedir. Bu bileşenler, MARL algoritmalarının güvenilir, ölçeklenebilir ve tekrarlanabilir değerlendirilmesini mümkün kılar.

Referanslar

[1] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun ve Shimon Whiteson. Starcraft çok ajanlı zorluğunda bağımsız öğrenme tek başına yeterli midir? *arXiv ön baskısı arXiv:2011.09533* , 2020.

[2] Matteo Bettini, Ajay Shankar ve Amanda Prorok. Heterojen çok robotlu takviyeli öğrenme. In *AA-MAS* , 2023.

[3] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen ve Yi Wu. İş birliği gerektiren çok etmenli oyunlarda PPO'nun şaşırtıcı etkinliği. *Sinirsel bilgi işleme sistemlerindeki ilerlemeler* , 35:24611–24624, 2022.

[4] Mehul Damani, Zhiyao Luo, Emerson Wenzel ve Guillaume Sartoretti. Primal_2 : Takviyeli ve taklit tabanlı çok etmenli öğrenmeyle yol bulma - yaşam boyu. *IEEE Robotik ve Otomasyon Mektupları*, 6(2):2666–2673, 2021.

[5] Alexey Skrynnik, Anton Andreychuk, Maria Nesterova, Konstantin Yakovlev ve Aleksandr Panov. Öğrenerek takip et: Planlama ve öğrenme yoluyla merkeziyetsiz, yaşam boyu çok etmenli yol bulma. In *Proceedings of the AAAI Conference on Artificial Intelligence* , cilt 38, sayfa 17541–17549, 2024.

[6] Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov ve Alexey Skrynnik. Mapf-gpt: Çok etmenli yol bulma için ölçekli taklit öğrenme. In *Proceedings of the AAAI Conference on Artificial Intelligence* , cilt 39, sayfa 23126–23134, 2025.

[7] Brian Angulo, Aleksandr Panov ve Konstantin Yakovlev . Dinamik engellerle yol planlamada uyarlanabilir hareket ilkel öğrenimi için politika optimizasyonu. *IEEE Robotics and Automation Letters*, 8(2):824–831, 2022.

[8] Weinan Chen, Wenzheng Chi, Sehua Ji, Hanjing Ye, Jie Liu, Yunjie Jia, Jiajie Yu, and Jiyu Cheng. A survey of autonomous robots and multi-robot navigation: Perception, planning and collaboration. *Biomimetic Intelligence and Robotics*, page 100203, 2024.

[9] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.

[10] Vassilissa Lehoux-Lebacque, Tomi Silander, Christelle Loiodice, Seungjoon Lee, Albert Wang, and Sofia Michel. Multi-agent path finding with real robot dynamics and interdependent tasks for automated warehouses. In *ECAI 2024*, pages 4393–4401. IOS Press, 2024.

[11] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

[12] Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr I Panov. Pogema: partially observable grid environment for multiple agents. *arXiv preprint arXiv:2206.10944*, 2022.

[13] Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin Yakovlev, and Aleksandr Panov. Pogema: A benchmark platform for cooperative multi-agent pathfinding. In *The Thirteenth International Conference on Learning Representations*, 2025.

[14] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 42–56. Springer, 2022.

[15] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

[16] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[17] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

[18] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[19] Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I Midgley, Elshadai Tegegn, Tristan Kalloniatis, et al. Jumanji: a diverse suite of scalable reinforcement learning environments in jax. *arXiv preprint arXiv:2306.09884*, 2023.

[20] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[21] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. Ieee, 2004.

[22] Olivier Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.

[23] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6:271–295, 2012.

[24] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[25] Joseph Suarez. Pufferlib: Making reinforcement learning libraries and environments play nice. *arXiv preprint arXiv:2406.12905*, 2024.

[26] Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[27] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35:3962–3974, 2022.

[28] Jingtian Yan, Zhifei Li, William Kang, Yulun Zhang, Stephen Smith, and Jiaoyang Li. Advancing mapf towards the real world: A scalable multi-agent realistic testbed (smart). *arXiv preprint arXiv:2503.04798*, 2025.

[29] Nathan R Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012.

[30] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. *GitHub repository*, 2018.

[8] Weinan Chen, Wenzheng Chi, Sehua Ji, Hanjing Ye, Jie Liu, Yunjie Jia, Jiajie Yu ve Jiyu Cheng. Otonom robotlar ve çok robotlu navigasyon üzerine bir inceleme: Algılama, planlama ve iş birliği. *Biyomimetik Zeka ve Robotik* , sayfa 100203, 2024.

[9] Binyu Wang, Zhe Liu, Qingbiao Li ve Amanda Prorok . Dinamik ortamlarda küresel rehberli takviyeli öğrenme ile mobil robot yol planlaması. *IEEE Robotics and Automation Letters* , 5(4):6932–6939, 2020.

[10] Vassilissa Lehoux-Lebacque, Tomi Silander, Christelle Loiodice, Seungjoon Lee, Albert Wang ve Sofia Michel. Otomatik depolar için gerçek robot dinamiği ve bağımlı görevlerle çok ajanlı yol bulma. ECAI 2024'te, sayfalar 4393–4401. IOS Press, *2024*.

[11] Georgios Papoudakis, Filippos Christianos, Lukas Schafer ve Stefano V. Albrecht. Kooperatif görevlerde çok ajanlı derin takviyeli öğrenme algoritmalarının karşılaştırma ölçütü. *arXiv preprint arXiv:2006.07869* , 2020.

[12] Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev ve Aleksandr I Panov. Pogema: Çoklu ajanlar için kısmen gözlemlenebilir ızgara ortamı. *arXiv preprint arXiv:2206.10944* , 2022.

[13] Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov , Alexander Chernyavskiy, Konstantin Yakovlev ve Aleksandr Panov. Pogema: Kooperatif çok etmenli yol bulma için bir karşılaştırma ölçütü platformu. The Thirteenth International Conference on Learning Representations , 2025.

[14] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp ve Amanda Prorok. Vmas: Kollektif robot öğrenimi için vektörleştirilmiş çok etmenli simülatör. Uluslararası Dağıtılmış Otonom Robotik Sistemler Sempozyumu , sayfa 42–56. Springer, 2022.

[15] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster ve Shimon Whiteson. Smacv2: İşbirlikçi çok ajanlı takviyeli öğrenme için geliştirilmiş bir karşılaştırma ölçütü. *Neural Information Processing Systems'ta Gelişmeler* , 36:37567–37593, 2023.

[16] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel ve Igor Mordatch. Karışık işbirlikçi-rekabetçi ortamlar için çok ajanlı aktör-eleştirmen. *Neural Information Processing Systems'ta Gelişmeler* , 30, 2017.

[17] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly ve diğerleri. Jaxmarl: Jax'ta çok ajanlı takviyeli öğrenme ortamları. *arXiv ön baskısı arXiv:2311.10090* , 2023.

[18] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster ve Shimon Whiteson. Starcraft çok ajanlı meydan okuması. *arXiv preprint arXiv :1902.04043* , 2019.

[19] Clement Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I Midgley, Elshadai Tegegn, Tristan Kalloniatis ve diğerleri. Jumanji: JAX'te ölçeklenebilir takviyeli öğrenme ortamlarından oluşan çeşitli bir paket. *arXiv preprint arXiv:2306.09884* , 2023

[20] Igor Mordatch ve Pieter Abbeel. Çok ajanlı popülasyonlarda temellendirilmiş bileşimsel dilin ortaya çıkışı. *arXiv önbaskısı arXiv:1703.04908* , 2017.

[21] *Nathan Koenig ve Andrew Howard. Gazebo için tasarım ve kullanım paradigmaları, açık kaynaklı çoklu robot simülatörü. 2004 IEEE/RSJ Uluslararası Akıllı Robotlar ve Sistemler Konferansı (IROS) (IEEE Cat. No. 04CH37566), cilt 3, s.* 2149–2154. *IEEE, 2004.*

[22] Olivier Michel. Cyberbotics ltd. webots™: profesyonel mobil robot simülasyonu. International Journal of Advanced Robotic Systems, 1(1):5, 2004.

[23] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle ve diğerleri. Argos: modüler, paralel, çok motorlu, çoklu robot sistemleri için simülatör. *Sürü zekası* , 6:271–295, 2012.

[24] *Jayesh K Gupta, Maxim Egorov ve Mykel Kochenderfer. Derin takviyeli öğrenme kullanarak işbirlikçi çok ajanlı kontrol. Autonomous Agents and Multi-agent Systems: AAMAS 2017 Atölyeleri, En İyi Makaleler, Sao Paulo, Brezilya, 8-12 Mayıs 2017, Revize Edilmiş Seçilmiş Makaleler 16* , *ss*. 66–83. Springer, *2017*.

[25] Joseph Suarez. Pufferlib: Takviyeli öğrenme kütüphaneleri ve ortamlarının uyumlu çalışmasını sağlama. *arXiv ön baskısı arXiv:2406.12905* , 2024.

[26] *Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes ve Jakob Nicolaus Foerster. Pişmanlık yok: Müfredat keşfi için pişmanlık yaklaşımlarını inceleme ve iyileştirme. Otuzsekizinci Yıllık Sinirsel Bilgi İşleme Sistemleri Konferansında , 2024.*

[27] Eugene Vinitsky, Nathan Lichtle, Xiaomeng Yang, Brandon Amos ve Jakob Foerster. Nocturne: çok etmenli öğrenmeyi gerçek dünyaya bir adım daha yaklaştıran ölçeklenebilir sürüş karşılaştırma ölçütü. *Sinirsel Bilgi İşleme Sistemlerindeki Gelişmeler* , 35:3962–3974, 2022.

[28] Jingtian Yan, Zhifei Li, William Kang, Yulun Zhang, Stephen Smith ve Jiaoyang Li. Gerçek dünyaya yönelik çok etmenli yol bulmayı ilerletmek: Ölçeklenebilir çok etmenli gerçekçi test ortamı (SMART). *arXiv ön baskısı arXiv:2503.04798* , 2025.

[29] Nathan R Sturtevant. Izgara tabanlı yol bulma için karşılaştırma ölçütleri. *IEEE Hesaplamalı Zekâ ve Oyunlarda Yapay Zekâ İşlemleri* , 4(2):144–148, 2012.

[30] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin , George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne ve diğerleri. Jax: Python + numpy programlarının bileşik dönüşümleri. *GitHub deposu* , 2018.

[31] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[32] Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl. *Advances in Neural Information Processing Systems*, 35:5510–5521, 2022.

[33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[34] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[35] Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. Torchrl: A data-driven decision-making library for pytorch, 2023.

[36] Matteo Bettini, Amanda Prorok, and Vincent Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024.

Appendix Contents

Appendix A — Evaluation Protocol

Evaluation Suite Overview. To make results easy to compare and fully reproducible, CAMAR adopts the "Standardised Performance Evaluation Protocol for Cooperative MARL" by [32] and *extends* it to continuous multi-agent path-finding. We keep the core ideas, fixed training budgets, multiple random seeds, and strict uncertainty estimates, but add path-finding-specific stress-tests:

- **Axis 1: Agent count.** We vary the number of agents to see whether a method still works when the team grows.
- **Axis 2: Map difficulty.** We change obstacle density and map geometry to check if the learned policy still solves harder layouts.

Three difficulty tiers.

- **Easy** uses the same map, agent count, and obstacle parameters for training and testing. Only the random seeds that place starts and goals change. This tells us whether an algorithm can *solve* the problem at all.
- **Medium** trains on one map setting but evaluates on 12 variants that share the same map types while changing agent count and obstacle density. This probes generalisation within the same domain and includes testing generalization across agents.
- **Hard** trains on any user-chosen maps—except the MovingAI street set—and then tests on those unseen street maps with new agent counts. This is a near-real-world stress test.

Standardized Evaluation Suite for CAMAR

Input. A set of maps $\mathcal{M}$, task sets $\mathcal{T}_m$, and a pool of algorithms $\mathcal{A}$.

1. Default settings

- Training steps T: **2M** (off-policy) or **20M** (on-policy).
- Independent runs R: **3** seeds.
- Evaluation episodes E: **1 000** per interval.
- Evaluation intervals $\mathcal{I}$ (based on available resources): every **10K-100K** steps (off-policy) or **100K-1000K** steps (on-policy).

2. Metrics

- *Returns* G for sample-efficiency plots.
- *Success Rate* (SR), *Flowtime* (FT), *Makespan* (MS), and *Coordination* (CO).
- Per-task: mean G_t^a over E episodes with 95% CIs.
- Per-protocol: build an $(R \times |\mathcal{T}|)$ matrix of normalised returns, then report IQM, optimality gap, probability of improvement, and performance profiles (all with 95% bootstrap CIs).

3. Three difficulty tiers

1. **Easy**: train and test on the same map/agent count. 12 models $\times$ 3 seeds.
2. **Medium**: train on one map, test on all 12 maps and aggregate. 12 models $\times$ 3 seeds.
3. **Hard**: train on any maps (except MovingAI), test on MovingAI [29] street maps with new agent counts. Single model (preferably 3 seeds).

4. Reporting checklist

- Hyper-parameters, network sizes, and compute budget.
- Map generation settings for each tier.
- Final IQM $\pm$95% CI for SR, FT, MS, CO (mandatory). Sample-efficiency curves are recommended for Easy and Medium, optional for Hard. In Hard also plot each metric against the number of agents to test the ability to scale.

Each tier follows the default budget: 2M steps for off-policy and 20M steps for on-policy algorithms, three random seeds, and evaluations every 10K-100K or 100K-1000K steps. We keep the JAX seed fixed at 5 and split it to generate all

[31] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Kuttler, Andrew Lefrancq, Simon Green, Víctor Valdes, Amir Sadik ve diğerleri. Deepmind lab. *tarXiv ön baskısı arXiv:1612.03801* , 2016.

[32] Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh ve Arnu Pretorius. Kooperatif çok etmenli pekiştirmeli öğrenme için standartlaştırılmış performans değerlendirme protokolüne doğru. *Advances in Neural Information Processing Systems* , 35:5510–5521, 2022.

[33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel ve Sergey Levine. Soft actor-critic: Stokastik bir aktör ile off-policy maksimum entropi derin takviyeli öğrenme. In *International conference on machine learning* , ss. 1861–1870. Pmlr, 2018.

[34] Steven LaValle. Rapidly-exploring random trees: Yol planlamada yeni bir araç. *Araştırma Raporu 9811* , 1998.

[35] Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis ve Vincent Moens. Torchrl: Pytorch için veri odaklı karar verme kütüphanesi, 2023.

[36] Matteo Bettini, Amanda Prorok ve Vincent Moens. Karşılaştırma Ölçütü: Çok Etmenli Takviyeli Öğrenmenin Karşılaştırılması. *Journal of Machine Learning Research* , 25(217):1–10, 2024.

Ek İçerikleri

Ek A — Değerlendirme Protokolü

Değerlendirme Paketi Genel Bakışı. Sonuçların kolay karşılaştırılabilir ve tamamen tekrarlanabilir olması için, CAMAR [32] tarafından önerilen "Kooperatif Çok Etmenli Takviyeli Öğrenme için Standartlaştırılmış Performans Değerlendirme Protokolü"nü benimser ve bunu sürekli çok etmenli yol bulma alanına uygular. Ana fikirleri, sabit eğitim bütçelerini, çoklu rastgele tohumları ve kesin belirsizlik tahminlerini koruyoruz, ancak yol bulmaya özgü stres testleri ekliyoruz:

- Eksen **1:** Ajan sayısı. Bir yöntemin takım büyüdüğünde hâlâ çalışıp çalışmadığını görmek için ajan sayısını değiştiriyoruz.
- Eksen **2:** Harita zorluğu. Öğrenilen politikanın daha zor düzenlemeleri çözüp çözmediğini kontrol etmek için engel yoğunluğu ve harita geometrisi değiştirilir.

Üç zorluk seviyesi.

- Kolay aynı harita, ajan sayısı ve engel parametreleri ile eğitim ve test gerçekleştirilir. Sadece başlangıç ve hedeflerin yerleşimini belirleyen rastgele tohumlar değişir. Bu, bir algoritmanın problemi tamamen çözüp çözmediğini gösterir.
- Orta bir harita ayarında eğitim verilir, ancak ajan sayısı ve engel yoğunluğu değişen ve aynı harita tiplerini paylaşan 12 varyant üzerinde değerlendirme yapılır. Bu, aynı alan içindeki genelleştirmeyi test eder ve ajanlar arası genelleştirmeyi içerir.
- Zor Kullanıcı tarafından seçilen herhangi bir haritada - MovinGAI sokak seti hariç - eğitim verilir ve ardından yeni ajan sayılarıyla o görülmemiş sokak haritalarında test edilir. Bu, gerçek dünyaya yakın bir stres testidir.

CAMAR için Standartlaştırılmış Değerlendirme Seti

Girdi. Bir harita seti $\mathcal{M}$, görev setleri $\mathcal{T}_m$ ve algoritma havuzu $\mathcal{A}$.

1. Varsayılan ayarlar

- Eğitim adımları T: **2M**(çevrimdışı) veya **20M**(çevrimiçi).
- Bağımsız denemeler R: **3**tohum .
- Değerlendirme bölümleri E:**1.000**her aralıkta.
- Değerlendirme aralıkları $\mathcal{I}$(mevcut kaynaklara bağlı olarak): her **10K-100K** adım (çevrimdışı) veya **100K-1000K** adım (çevrimiçi).

2. Ölçütler

- Ödüller Görnek verimliliği grafiklerinde.
- *Başarı Oranı* (SR), *Akış Süresi* (FT), *Tamamlama Süresi (MS) ve Koordinasyon* (*KO*).
- Görev başına: ortalama G E bölümlerinde %95 güven aralıkları ile.
- Protokol başına: normalleştirilmiş ödüllerden oluşan ($R \times |\mathcal{T}|$) matrisi oluşturulur, ardından IQM, optimalite farkı , iyileşme olasılığı ve performans profilleri (tümü %95 bootstrap güven aralıklarıyla) raporlanır.

3. Üç zorluk seviyesi

1. Kolay : aynı harita/ajan sayısı üzerinde eğitim ve test. 12 model $\times$ 3 tohum.
2. Orta : bir harita üzerinde eğitim, tüm 12 haritada test ve birleştirme. 12 model $\times$ 3 tohum.
3. Zor : herhangi bir harita üzerinde eğitim (MovingAI hariç), MovingAI [29] sokak haritalarında yeni ajan sayıları ile test. Tek model (tercihen 3 tohum).

4. Raporlama kontrol listesi

- Hiperparametreler, ağ büyüklükleri ve hesaplama bütçesi.
- Her kademeye ait harita oluşturma ayarları.
- Son IQM $\pm$%95 güven aralığı için SR, FT, MS, KO (zorunlu). Örnek verimliliği eğrileri Kolay ve Orta için önerilir, Zor için isteğe bağlıdır. Zor kategoride ayrıca ölçeklenebilirliği test etmek amacıyla her metriğin ajan sayısına karşı grafiği çizilir.

Her kademede varsayılan bütçe uygulanır: politika dışı algoritmalar için 2M adım ve politika içi algoritmalar için 20M adım, üç rastgele tohum ve her 10K-100K veya 100K-1000K adımdan sonra değerlendirme. JAX tohumunu sabit tutuyoruz 5ve tümünü oluşturmak için bölüyoruz

	Easy	Medium	Hard
Purpose	Test that the method can solve the problem without testing generalization.	Test that the method can solve the problem and generalise across similar map types including varying number of agents.	Test that the method can generalise to near-real-world settings.
How to train and evaluate	**Train on** `map_X` (see Appendix). **Evaluate on** *the same* `map_X`. **Repeat** for all 12 maps	**Train on** `map_X` (see Appendix). **Evaluate on** *all 12* maps. **Repeat** for all 12 maps	**Train on** any map collection excluding MovingAI **Evaluate on** MovingAI street collection with varying agent counts
Number of models **Number of evals**	12 trained models × 3 seeds 12 evaluations × 1K episodes	12 trained models × 3 seeds 144 evaluations × 1K episodes	1 trained model 30 evaluations × 1K episodes × varying number of agents
Report	**Metrics** - if needed **sample-efficiency curves** - strongly recommended	**Metrics** - mandatory **sample-efficiency curves** - if needed	**Metrics** - mandatory **sample-efficiency curves** - if resources allow **metrics vs num_agents** - mandatory

Table 3: Overview on 3-tier evaluation protocols presenting the purpose, number of trained models and evaluations for each protocol

evaluation keys, which makes every run exactly repeatable.

Reporting and analysis. We require final IQM ± 95% CI of Success Rate, Flowtime, Makespan, and Coordination. Sample-efficiency curves are strongly recommended for Easy and Medium and optional for Hard if resources allow. For the Hard tier, we also ask for plots of each metric versus the number of agents because scalability is a key question.

Appendix B — Code Examples

The CAMAR library provides a simple and flexible interface for building custom multi-agent pathfinding environments. Below we show two examples that demonstrate how to create environments using different map generators and agent dynamics.

Example 1 (Fig. 9) shows how to create an environment with the `random_grid` map. It uses the string-based API to pass all configuration options directly. This includes the number of agents, the size range for agents and goals, and the dynamic model to use. We set the dynamic to `HolonomicDynamic`, and also customize the observation window and the shaping factor. This example shows how to use heterogeneous sizes for agents and goals, which can help simulate more realistic scenarios.

Example 2 (Fig. 10) shows how to use maps from the MovingAI benchmark [29]. These maps are loaded by name using the map generator function movingai(). This example also shows how to use heterogeneous agent dynamics by combining different models (e.g. `DiffDriveDynamic` and `HolonomicDynamic`). This is done using the class-based API with MixedDynamic. The number of agents of each type is specified, and the total is passed to the environment. This allows testing how different types of agents can cooperate in complex environments.

```python
from camar import camar_v0

env1 = camar_v0(
    map_generator="random_grid",
    dynamic="HolonomicDynamic",
    pos_shaping_factor=2.0,
    window=0.25, # obs window
    max_obs=10, # max num of obj in obs
    map_kwargs={
        "num_agents": 16,
        "agent_rad_range": (0.01,  0.05 ),
        "goal_rad_range":  (0.001, 0.005),
    },
    dynamic_kwargs={
        "mass": 2.0,
    }
)
```

Figure 9: Example 1. Creating a CAMAR environment using random grid maps and holonomic agents. The environment is loaded using the string-based API, which supports configuration via YAML or inline dictionary.

These examples can be used as templates for building new scenarios in CAMAR. Users can adjust map parameters, agent settings, or dynamic models to suit their research needs. The CAMAR design supports quick changes to both environment structure and agent behavior using only a few lines of code.

Appendix C — Limitations

CAMAR offers fast and flexible simulation, but there are still important gaps. First, all built-in scenarios use static obstacles. We do not yet provide maps with moving obstacles, so agents cannot train on fully dynamic scenes. While the engine

	Kolay	Orta	Zor
Amaç	Yöntemin problemi çözüp çözmediğini test edin ancak genelleme testi yapmayın.	Yöntemin problemi çözüp çözemediğini ve farklı ajan sayıları dahil benzer harita türlerinde genelleme yapabildiğini test edin.	Yöntemin gerçeğe yakın ortamlara genelleme yapabildiğini test edin.
Nasıl eğitilir ve değerlendirilir	Harita X üzerinde eğitim yapın (Bkz. Ek). Aynı harita X üzerinde değerlendirin. Tüm 12 harita için tekrarlayın	Harita X üzerinde eğitim yapın (Bkz. Ek). Tüm *12* harita üzerinde değerlendirin. Tüm 12 harita için tekrarlayın	MovingAI dışındaki herhangi bir harita koleksiyonunda eğitim yapın Ajan sayısı değişen MovingAI sokak koleksiyonu üzerinde değerlendirin
Model sayısı Değerlendirme sayısı	12 eğitimli model ×3 tohum 12 değerlendirme ×1K bölüm	12 eğitimli model ×3 tohum 144 değerlendirme × 1K bölüm	1 eğitimli model 30 değerlendirme × 1K bölüm × değişken ajan sayısı
Rapor	Metrikler- gerekiyorsa örnek-verimlilik eğrileri - şiddetle önerilir	Metrikler- zorunlu örnek verimliliği eğrileri - gerekirse	Metrikler- zorunlu örnek verimliliği eğrileri - kaynaklar izin verirse metrikler ve ajan_sayısı - zorunlu

Tablo 3: Her protokol için amaç, eğitilen model sayısı ve değerlendirmeleri gösteren 3 kademeli değerlendirme protokolleri özeti

değerlendirme anahtarları, her çalışmanın tam olarak tekrarlanabilir olmasını sağlar.

Raporlama ve analiz. Başarı Oranı, Akış Süresi, Tamamlama Süresi ve Koordinasyon için son IQM ± %95 GA gereklidir. Örnek-verimlilik eğrileri Kolay ve Orta için şiddetle önerilir; Zor için ise kaynaklar izin verirse isteğe bağlıdır. Zor kademesi için, ölçeklenebilirlik temel bir konu olduğundan her metrikle ajan sayısı arasındaki grafiklerin sunulması talep edilir.

Ek B — Kod Örnekleri

CAMAR kütüphanesi, özel çok etmenli yol bulma ortamları oluşturmak için basit ve esnek bir arayüz sunmaktadır. Aşağıda, farklı harita üreteçleri ve ajan dinamikleri kullanarak ortamların nasıl oluşturulacağını gösteren iki örnek verilmiştir.

Örnek **1** (Şekil 9), rastgele ızgara haritası ile bir ortamın nasıl oluşturulacağını göstermektedir. Tüm yapılandırma seçenekleri string tabanlı API aracılığıyla doğrudan aktarılmaktadır. Buna ajan sayısı, ajan ve hedeflerin boyut aralığı ile kullanılacak dinamik model dahildir. Dinamiği HolonomikDinamik olarak belirledik ve ayrıca gözlem penceresi ile şekillendirme faktörünü özelleştirdik. Bu örnek, daha gerçekçi senaryoların simülasyonu için ajanlar ve hedefler arasında heterojen boyutların nasıl kullanılacağını göstermektedir.

Örnek **2** (Şekil 10), MovingAI karşılaştırma ölçütünden [29] haritaların kullanımını göstermektedir. Bu haritalar, movingai () harita üreteç fonksiyonu kullanılarak isimleriyle yüklenmektedir. Bu örnek ayrıca, farklı modelleri (örneğin `DiffDriveDynamic` ve HolonomikDinamik) birleştirerek heterojen ajan dinamiklerinin nasıl kullanılacağını göstermektedir. Bu, MixedDynamic ile sınıf tabanlı API kullanılarak gerçekleştirilir. Her türden ajan sayısı belirtilir ve toplamı ortamına iletilir . Bu, farklı türdeki ajanların karmaşık ortamlarda nasıl işbirliği yapabileceğini test etmeye olanak tanır.

```python
from camar import camar_v0

env1 = camar_v0(
    map_generator="rastgele_ızgara",
    dynamic="HolonomikDinamik",
    pos_shaping_factor=2.0,
    window=0.25,   # gözlem penceresi
    max_obs=10,   # gözlemdeki maksimum nesne sayısı
    map_kwargs={
        "num_agents": 16,
        "agent_rad_range": (0.01,  0.05 ),
        "goal_rad_range":  (0.001, 0.005),
    },
    dynamic_kwargs={
        "mass": 2.0,
    }
)
```

Şekil 9: Örnek 1. Rastgele ızgara haritaları ve holonomik ajanlar kullanılarak CAMAR ortamı oluşturulması. Ortam, YAML veya satır içi sözlükle yapılandırmayı destekleyen string tabanlı API kullanılarak yüklenir.

Bu örnekler, CAMAR'da yeni senaryolar oluşturmak için şablon olarak kullanılabilir. Kullanıcılar araştırma ihtiyaçlarına göre harita parametrelerini, ajan ayarlarını veya dinamik modelleri değiştirebilir. CAMAR tasarımı, sadece birkaç satır kodla ortam yapısında ve ajan davranışlarında hızlı değişikliklere imkan tanır.

Ek C — Sınırlılıklar

CAMAR hızlı ve esnek simülasyon sağlar; ancak hâlen önemli eksiklikler mevcuttur. Öncelikle, tüm hazır senaryolar statik engeller kullanmaktadır . Henüz hareketli engelli haritalar sunmuyoruz, dolayısıyla ajanlar tam anlamıyla dinamik sahnelerde eğitim yapamamaktadır. Motor...

```python
from camar import camar_v0
from camar.maps import movingai
from camar.dynamics import MixedDynamic,
    DiffDriveDynamic, HolonomicDynamic

dynamics_batch = [
    DiffDriveDynamic(mass=1.0),
    HolonomicDynamic(mass=10.0),
]
# 8 diffdrive + 24 holonomic = 32 total
num_agents_batch = [8, 24]
mixed_dynamic = MixedDynamic(
    dynamics_batch=dynamics_batch,
    num_agents_batch=num_agents_batch,
)

map_generator = movingai(
    map_names=["street/Denver_0_1024",
               "bg_maps/AR0072SR"],
    num_agents=sum(num_agents_batch),
)

env2 = camar_v0(
    map_generator=map_generator,
    dynamic=mixed_dynamic,
    pos_shaping_factor=2.0,
    window=0.25,
    max_obs=10,
)
```

Figure 10: Example 2. Creating a CAMAR environment with MovingAI maps [29] and mixed agent dynamics. This example demonstrates the class-based API for explicitly specifying heterogeneous agent behavior.

supports different sizes for agents, goals, and landmarks, the built-in map generators do not include variations in landmark sizes. We believe that adding new maps with obstacles of different sizes and dynamic behavior is an important direction for future work. Fortunately, CAMAR's modular design makes it easy to extend in this way.

Second, although we support two built-in dynamics models (`HolonomicDynamic` and `DiffDriveDynamic`), and allow mixing them to build heterogeneous teams, both use basic integration schemes: either explicit or non-explicit Euler. To achieve stable simulation with larger time steps (for example, dt = 0.1), a frameskip must be applied. Without frameskip, simulation becomes unstable with large dt values, and a smaller integration step like dt = 0.005 must be used together with frameskip = 20. Adding more stable integration methods, such as Runge-Kutta, could improve stability and allow efficient simulation with fewer steps. For example, using Runge-Kutta with dt = 0.05 and frameskip = 2 (for a total of 8 integration steps per forward pass) might offer a better trade-off between speed and accuracy.

Third, there is no built-in communication mechanism between agents. Adding message passing would help study how algorithms use shared information.

Appendix D — Extended Benchmark Performance Analysis

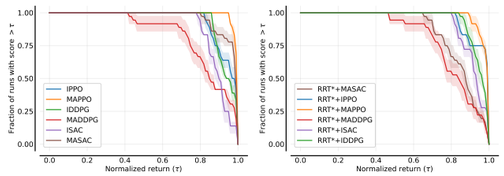


Figure 11: Performance profiles based on normalized return. Each line shows the fraction of evaluation runs that achieved a score above a given threshold τ.

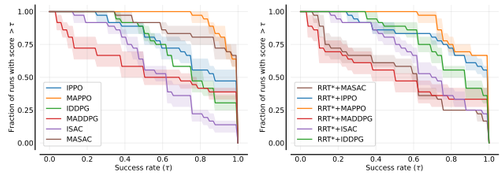


Figure 12: Performance profiles based on success rate. Each line shows the fraction of evaluation runs that achieved SR above a threshold τ.

Aggregate Analysis and Performance Profiles We also report aggregate return metrics across all evaluated methods in Fig.13. These charts, produced using the MARL-EVAL toolkit[32], show median, interquartile mean (IQM), mean return, and optimality gap, with normalized scores. Among MARL baselines, MAPPO performs best across all metrics, followed by IPPO. ISAC and IDDPG achieve lower scores and are less consistent, while MADDPG performs the worst and shows the largest optimality gap. Among hybrid methods, RRT*+MAPPO and RRT*+IPPO achieve strong results that match or exceed their baselines. However, RRT*+MASAC and RRT*+MADDPG perform poorly. This may be caused by the larger input space, especially for centralized critics that struggle with high-dimensional observations.

To further evaluate consistency and reliability, we present performance profiles in Fig.11 and Fig.12. These plots show the fraction of evaluation runs where each algorithm achieves a score above a given threshold. The results confirm our earlier findings: MAPPO and IPPO maintain good performance across tasks, while MADDPG and MASAC drop quickly. RRT*+MAPPO and RRT*+IPPO remain strong among hybrid approaches, while RRT*+MADDPG and RRT*+MASAC again show poor results.

These charts together highlight the differences in generalization ability, robustness, and effectiveness across MARL and hybrid methods. They also support our earlier analysis of SR, FT, MS, CO (Table 2). Overall, these findings show that RRT* integration can improve performance for some methods, but not all. The effect depends on the algorithm and how it handles the added input complexity.

```python
from camar import camar_v0
from camar.maps import movingai
from camar.dynamics import MixedDynamic,
    DiffDriveDynamic, HolonomicDynamic

dynamics_batch = [
    DiffDriveDynamic(mass=1.0),
    HolonomicDynamic(mass=10.0),
]
# 8 diffdrive + 24 holonomik = toplam 32
num_agents_batch = [8, 24]
mixed_dynamic = MixedDynamic(
    dynamics_batch=dynamics_batch,
    num_agents_batch=num_agents_batch,
)

map_generator = movingai(
    map_names=["street/Denver_0_1024",
               "bg_maps/AR0072SR"],
    num_agents=  toplam(num_agents_batch),
)

env2 = camar_v0(
    map_generator=map_generator,
    dynamic=mixed_dynamic,
    pos_shaping_factor=2.0,
    window=0.25,
    max_obs=10,
)
```

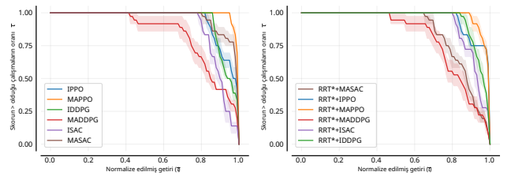
Şekil 10: Örnek 2. MovingAI haritaları [29] ve karma ajan dinamikleri kullanılarak bir CAMAR ortamı oluşturulması. Bu örnek, heterojen ajan davranışlarını açıkça belirtmek için sınıf tabanlı API'nin kullanımını gösterir.

Ajanlar, hedefler ve işaret noktaları için farklı boyutları desteklese de, yerleşik harita oluşturucular işaret noktası boyutu çeşitliliği sunmamaktadır. Farklı boyutlarda engeller ve dinamik davranış içeren yeni haritaların eklenmesi, gelecekteki çalışmalar için önemli bir yön olarak değerlendirilmektedir. Neyse ki, CAMAR'ın modüler tasarımı bu tür genişletmeleri kolaylaştırmaktadır.

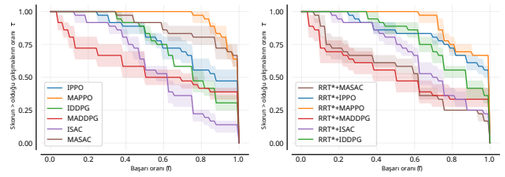
İkincisi, iki yerleşik dinamik modeli (HolonomikDinamik ve `DiffDriveDynamic`) desteklememize rağmen ve heterojen takımlar oluşturmak için bunların karıştırılmasına izin vermemize rağmen, her ikisi de temel entegrasyon şemaları kullanır: açık veya kapalı Euler. Daha büyük zaman adımlarında (örneğin , dt = 0.1) stabil simülasyon sağlamak için frameskip uygulanmalıdır. Frameskip olmadan, simülasyon büyük dt değerlerinde kararsız hale gelir ve frameskip = 20 ile birlikte dt = 0.005 gibi daha küçük bir entegrasyon adımı kullanılmalıdır. Runge-Kutta gibi daha stabil entegrasyon yöntemlerinin eklenmesi, stabiliteyi artırabilir ve daha az adımda verimli simülasyona olanak tanıyabilir. Örneğin, dt = 0.05 ve frameskip = 2 (her ileri geçişte toplam 8 entegrasyon adımı) ile Runge-Kutta kullanmak, hız ve doğruluk arasında daha iyi bir denge sağlayabilir.

Üçüncü olarak, ajanlar arasında yerleşik bir iletişim mekanizması bulunmamaktadır. Mesaj iletiminin eklenmesi, algoritmaların paylaşılan bilgiyi nasıl kullandığını incelemeyi kolaylaştırır.

Ek D — Genişletilmiş Karşılaştırma Ölçütü Performans Analizi



Şekil 11: Normalize edilmiş getiriler temel alınarak oluşturulan performans profilleri. Her satır, değerlendirme çalışmalarından belirtilen eşik τ değerinin üzerinde skor elde edenlerin oranını gösterir.



Şekil 12: Başarı oranına dayalı performans profilleri. Her satır , değerlendirme çalışmalarında başarı oranının belirtilen eşik τ değerinin üzerinde olduğu oranı gösterir.

Toplu Analiz ve Performans Profilleri Ayrıca, tüm değerlendirilmiş yöntemlere ait toplu getiri metriklerini Şekil 13'te sunuyoruz. MARL-EVAL araç seti[32] kullanılarak oluşturulan bu grafikler normalize skorlar ile medyan, çeyrekler arası ortalama (IQM), ortalama getiri ve optimalite farkını göstermektedir. MARL karşılaştırma ölçütleri arasında, tüm metriklerde MAPPO en iyi performansı göstermekte olup, bunu IPPO takip etmektedir. ISAC ve IDDPG daha düşük skorlar almakta ve tutarlılığı daha azdır; MADDPG ise en düşük performansı göstererek en büyük optimalite farkını ortaya koymaktadır. Karma yöntemler arasında ise RRT*+MAPPO ve RRT*+IPPO güçlü sonuçlar vererek temel yöntemleriyle eşdeğer veya daha iyi performans sergilemektedir. Ancak, RRT*+MASAC ve RRT*+MADDPG zayıf performans göstermektedir. Bu durum , özellikle yüksek boyutlu gözlemlerle başa çıkmakta zorlanan merkezi eleştirmenler için daha geniş giriş alanından kaynaklanıyor olabilir.

Tutarlılık ve güvenilirliği daha ayrıntılı değerlendirmek üzere, Şekil 11 ve Şekil 12'de performans profillerini sunuyoruz. Bu grafikler, her algoritmanın belirli bir eşik değerin üzerindeki skoru elde ettiği değerlendirme denemelerinin oranını göstermektedir. Sonuçlar, önceki bulgularımızı teyit etmektedir: MAPPO ve IPPO görevler genelinde iyi performansını sürdürürken, MADDPG ve MASAC hızla düşüş yaşamaktadır. RRT*+MAPPO ve RRT*+IPPO karma yaklaşımlar arasında güçlü konumlarını korurken, RRT*+MADDPG ve RRT*+MASAC yine zayıf sonuçlar vermektedir.

Bu grafikler bir arada, MARL ve karma yöntemler arasındaki genelleme yeteneği, dayanıklılık ve etkinlik farklılıklarını ortaya koymaktadır. Ayrıca, Tablo 2'deki SR, FT, MS, KO değerleri önceki analizimizi desteklemektedir. Genel olarak, bu bulgular RRT* entegrasyonunun bazı yöntemlerde performansı artırabileceğini ancak hepsinde artırmayacağını göstermektedir. Etki, algoritmaya ve eklenen girdi karmaşıklığının nasıl işlendiğine bağlıdır.

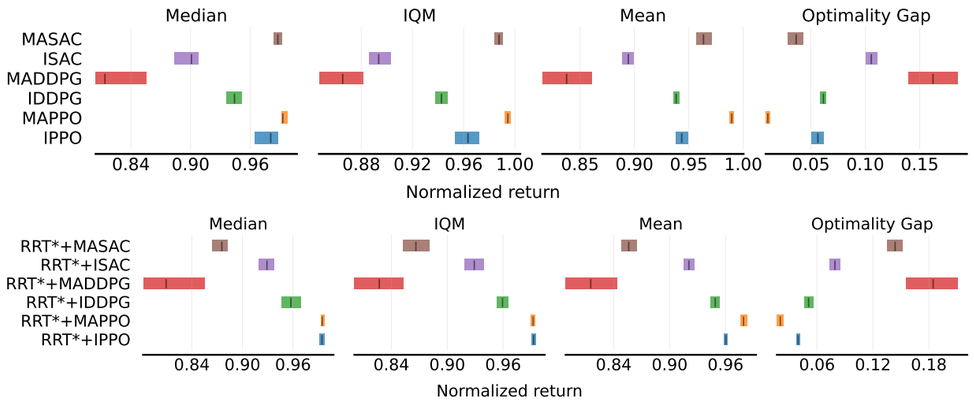


Figure 13: Normalized return scores. Top: MARL baselines. Bottom: RRT* hybrid methods. Metrics include median, interquartile mean (IQM), mean, and optimality gap. Higher values are better except for optimality gap. Each bar shows 95% confidence intervals.

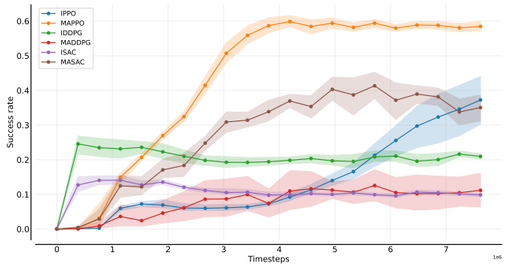


Figure 14: Sample-efficiency curves: mean success rate versus environment steps. Computed using MARL-EVAL.

Algorithm	SR ↑	FT ↓	MS ↓	CO ↑	Ep. Return ↑
IDDPG (SC)	0.77	406	134	0.996	266
ISAC (SC)	0.21	1060	160	0.999	92
MADDPG (SC)	0.00	1241	160	0.997	-1.1
MADDPG (NSC)	0.00	1259	160	0.998	-0.9
MASAC (SC)	0.26	813	159	0.996	67
MASAC (NSC)	0.44	780	160	1.000	193

Table 4: Performance comparison of multi-agent DDPG and SAC variants on `random_grid` with 8 agents. SC: shared critic, NSC: not shared critic.

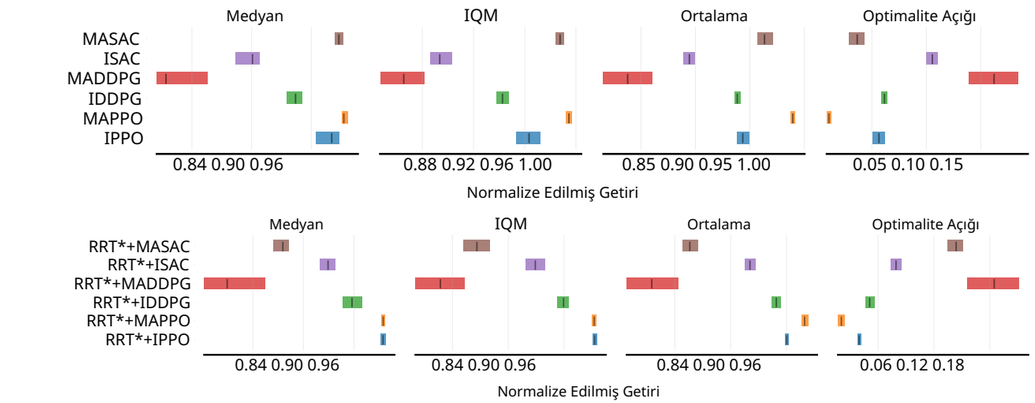
Sample Efficiency. Figure 14 shows the mean success rate evolving during training, computed with the MARL-EVAL toolkit [32]. MAPPO learns the fastest, rising above 0.60 after roughly $3M$ steps and then remaining stable. MASAC improves quickly at first but plateaus below 0.45. IPPO starts slowly yet keeps improving and nearly reaches the MASAC curve near the end of training. IDDPG attains a modest plateau early and then changes little. ISAC climbs slightly in the first million steps and then flattens out, while MADDPG stays under 0.15 for the entire run.

DDPG and SAC analysis We further investigated the weak performance of multi-agent DDPG and SAC algorithms in our benchmark. Specifically, we observed that both MADDPG and MASAC achieved lower success rates and higher episode lengths compared to their independent counterparts (IDDPG and ISAC). To understand these results, we considered two possible factors that could negatively affect learning with centralized critics.

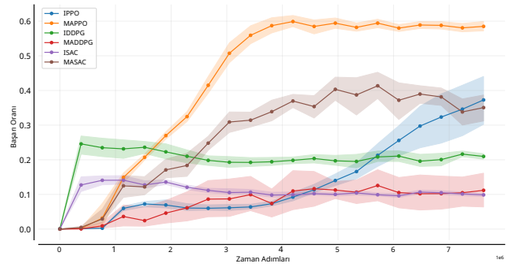
First, the credit assignment problem becomes harder as the number of agents increases. In both MADDPG and MASAC, a single centralized critic must estimate value based on the combined actions and states of all agents. The critic's target is a simple mean across agents in this version, which can blur important agent-specific differences. This makes learning slow and unstable.

Second, the input size of the centralized critic grows with the number of agents. For example, in our setting with 8 agents, the critic receives a long concatenated vector of all observations and actions. This high-dimensional input makes the learning problem more difficult, especially for relatively small networks like MLPs. Previously discussed performance of RRT*+MASAC and RRT*+MADDPG also supports this hypothesis. These hybrid methods show much lower returns and success rates compared to RRT*+IPPO or RRT*+MAPPO. This drop in performance is likely caused by the added RRT* information increasing the input size even further, making the learning task more difficult for centralized critics.

To test these ideas, we trained new versions of MADDPG and MASAC where the critic network is no longer shared between agents. Instead, each agent has its own critic (without parameter sharing), while still using centralized training.



Şekil 13: Normalize edilmiş getiri puanları. Üst: Çok Eylemli Pekiştirmeli Öğrenme taban çizgileri. Alt: RRT* karma yöntemleri. Ölçütler; medyan, çeyrekler arası ortalama (IQM), ortalama ve optimalite açığını içerir. Optimalite açığı hariç, daha yüksek değerler daha iyidir. Her çubuk %95 güven aralıklarını gösterir.



Şekil 14: Örnek verimliliği eğrileri: ortalama başarı oranı ile ortam adımları arasındaki ilişki. Hesaplama MARL - EVAL aracı kullanılarak yapılmıştır.

Algoritma	SR ↑	FT ↓	MS ↓	KO ↑	Ep. Dönüş ↑
IDDPG (KO)	0.77	406	134	0.996	266
ISAC (KO)	0.21	1060	160	0.999	92
MADDPG (KO)	0.00	1241	160	0.997	-1.1
MADDPG (KO olmayan)	0.00	1259	160	0.998	-0.9
MASAC (KO)	0.26	813	159	0.996	67
MASAC (KO olmayan)	0.44	780	160	1.000	193

Tablo 4: 8 ajanlı rastgele ızgarada çok ajanlı DDPG ve SAC varyantlarının performans karşılaştırması. KO: paylaşılan eleştirmen, KO olmayan: paylaşılan eleştirmen değil.

Örnek Verimliliği. Şekil 14, eğitim sürecinde gelişen ortalama başarı oranını göstermekte olup, MARL - EVAL araç seti [32] kullanılarak hesaplanmıştır. MAPPO yaklaşık 3 M adımda 0,60'ın üzerine çıkarak en hızlı öğrenen yöntem olur ve ardından stabil kalır. MASAC başlangıçta hızlı ilerler ancak 0,45'in altında bir platoya ulaşır. IPPO yavaş başlar ancak sürekli gelişir ve eğitimin sonunda MASAC eğrisine yaklaşır. IDDPG erken dönemde düşük bir platoya ulaşır ve sonrasında pek değişmez. ISAC ilk bir milyon adımda hafifçe yükselir, ardından yatay seyrederken MADDPG tüm süreç boyunca 0,15'in altında kalır.

DDPG ve SAC analizi Çok ajanlı DDPG ve SAC algoritmalarının karşılaştırma ölçütümüzdeki zayıf performanslarını daha detaylı inceledik. Özellikle, MADDPG ve MASAC'ın bağımsız muadilleri olan IDDPG ve ISAC'a kıyasla daha düşük başarı oranları ve daha uzun bölüm süreleri elde ettiğini gözlemledik. Bu sonuçları anlayabilmek için merkezi eleştirmenlerle yapılan öğrenmeyi olumsuz etkileyebilecek iki olası faktörü değerlendirdik.

Birincisi, ajan sayısı arttıkça kredi atama problemi daha karmaşık hâle gelir. Hem MADDPG hem de MASAC'da, Tek bir merkezi eleştirmen, tüm ajanların birleşik eylem ve durumlarına dayanarak değeri tahmin etmelidir. Bu sürümde eleştirmenin hedefi, ajanlar arasındaki basit bir ortalamadır; bu durum, önemli ajan özgü farklılıkları bulanıklaştırabilir. Bu durum, öğrenmeyi yavaş ve kararsız hale getirir.

İkinci olarak, merkezi eleştirmenin giriş boyutu, ajan sayısıyla birlikte artar. Örneğin, 8 ajanlı ortamımızda, eleştirmen tüm gözlemlerin ve eylemlerin uzun birleştirilmiş vektörünü alır. Bu yüksek boyutlu girdi, özellikle MLP gibi nispeten küçük ağlar için öğrenme problemini zorlaştırmaktadır. Önceden tartışılan RRT*+MASAC ve RRT*+MADDPG performansları da bu hipotezi desteklemektedir . Bu karma yöntemler, RRT*+IPPO veya RRT*+MAPPO ile karşılaştırıldığında çok daha düşük getiri ve başarı oranları göstermektedir. Performanstaki bu düşüşün nedeni muhtemelen eklenen RRT* bilgisinin girdi boyutunu daha da artırması ve böylece merkezi eleştirmenler için öğrenme görevini zorlaştırmasıdır.

Bu fikirleri test etmek amacıyla, eleştirmen ağlarının ajanlar arasında paylaşılmadığı yeni MADDPG ve MASAC versiyonları eğittik . Her ajan kendi eleştirmenine sahiptir (parametre paylaşımı olmadan), ancak merkezi eğitim kullanılmaya devam etmektedir.

The results are shown in Table 4. For MADDPG, there was no improvement: both the shared and non-shared versions failed to learn the task, with success rate remaining at 0.0. This further supports the hypothesis that MADDPG struggles with large input vectors and credit assignment, even when the critics are separated. In contrast, MASAC improved with non-shared critics: the success rate rose from 0.26 to 0.44 and return nearly tripled. This suggests that SAC can better tolerate large input sizes, likely due to its entropy regularization and smoother policy updates. However, even this improved MASAC variant still performs worse than ISAC, which does not face centralized credit assignment at all. This shows that credit assignment remains a challenge in MASAC, despite its robustness to large inputs. Notably, the poor performance of RRT*+MADDPG and RRT*+MASAC in earlier experiments aligns with these findings.

Together, these findings confirm that both factors—the size of the critic input and the difficulty of shared credit assignment—can reduce the performance of centralized off-policy MARL methods. We include these results in Table 4.

Benchmark Maps

CAMAR includes three benchmark map types designed to evaluate navigation and coordination in multi-agent scenarios. These maps differ in layout, complexity, and agent configurations.

`random_grid` map is a 20×20 grid with randomly placed rectangular obstacles (with circle-based discretization). Agents are initialized in free cells, and goals are placed at randomly chosen locations. The scenario uses holonomic dynamics and includes parameters for obstacle density, agent radius, goal radius, and others. During benchmarking, we generate six different `random_grid` instances: three maps with 8 agents and three with 32 agents. Each of these variations is evaluated under three different obstacle densities (0.0, 0.05, 0.15), which define how cluttered the map becomes. These tasks are relatively simple but still useful to test multi-agent pathfinding (MAPF) algorithms.

`labmaze_grid` is based on procedural maze generation. It creates more structured maps using multiple rectangular rooms and corridors. The parameter `extra_connection_probability` controls how many connections exist between rooms: 1.0 means the map is fully connected, while lower values introduce more isolated areas and bottlenecks. As with the random grid, we evaluate 6 instances: three maps with 8 agents and three with 32 agents, each under three different connection probabilities (0.4, 0.65, 1.0). The dynamics are the same as in the random grid, allowing consistent comparison across layouts. These tasks provide harder challenges for coordination and navigation.

These benchmark maps are scalable, fast to simulate, and allow flexible control over structure and difficulty. Together, they form a comprehensive testbed for MAPF.

Heterogeneous Support

The `hetero_give_way` map tests heterogeneous multi-agent coordination. It is based on a simple corridor scenario where one larger agent cannot enter the central chamber, while the smaller agent can pass through. This setup forces agents to learn implicit turn-taking or yielding behavior. Although the dynamics remain the same (`HolonomicDynamic`), this configuration highlights differences in agent capabilities. Basic algorithms like MAPPO and IPPO struggle in this task. Nonetheless, CAMAR supports such heterogeneous settings out of the box and can be used to advance research in coordination strategies for agents with diverse sizes, dynamics, and behaviors.

Appendix E — Extended Scalability Analysis

We also study CAMAR scalability in comparison to other popular MARL environments to better understand its limits.

Comparing with Other Environments. In Table 5, we compare CAMAR to other environments with GPU support like JaxMARL and Jumanji. CAMAR is faster than VMAS and the MPE baselines, and it reaches speeds close to the best GPU simulators. For example, CAMAR runs at 338K SPS with 4 agents and 50K SPS with 32 agents. Some JaxMARL environments reach higher throughput—for example, SMAX achieves up to 1.2M SPS—but their performance drops more quickly as the number of agents grows. JaxNav, for instance, cannot run with more than 32 agents. While SMAX and RWARE show strong throughput, they use simplified settings like open space or discrete grids and do not support procedurally generated obstacle layouts. CAMAR, in contrast, supports continuous control and dense obstacles, which brings more realism. It trades a small drop in speed for a richer environment that better matches real-world MAPF challenges.

Agents	4	8	16	32	64	128
CAMAR	338K	189K	99K	50K	25K	13K
VMAS	16K	10K	5.5K	2.7K	1.2K	447
JaxNav	159K	61K	13.6K	353	-	-
SMAX	0.8M	1.0M	1.2M	0.8M	0.3M	0.1M
RWARE	1.4M	1.1M	0.8M	0.5M	0.3M	0.1M
MPE-2	0.6M	0.8M	1.2M	0.7M	0.5M	0.2M
MPE-1120	11.8K	11.2K	11.9K	11.1K	9.7K	7.7K

Table 5: SPS vs number of agents. Comparison across environments with GPU support. MPE-2 uses the default simple-tag configuration with 2 circle obstacles, while MPE-1120 uses 1120 circles matching the number of circle obstacles in CAMAR. '-' indicates that the environment was too slow to even finish episode.

Appendix F — Future Work

We highlight several research directions that can be explored using CAMAR.

First, CAMAR allows testing with many agents, which supports the development of scalable multi-agent reinforcement learning (MARL) methods. Future work can focus on improving how algorithms handle hundreds of agents in complex environments.

Second, CAMAR is a good platform for studying communication between agents. In large-scale tasks, agents often

Sonuçlar Tablo 4'te gösterilmektedir. MADDPG için herhangi bir iyileşme gözlemlenmedi: hem paylaşılan hem de paylaşımlı olmayan versiyonlar görevi öğrenemedi ve başarı oranı 0,0 olarak kaldı. Bu, MADDPG'nin büyük giriş vektörleri ve kredi dağıtımı konusunda , eleştirmenler ayrı olsa bile zorlandığı hipotezini daha da desteklemektedir. Buna karşılık, MASAC paylaşılmayan eleştirmenlerle iyileşme göstermiştir: başarı oranı %0,26'dan %0,44'e yükselmiş ve getiri neredeyse üç kat artmıştır. Bu durum, SAC'nin muhtemelen entropi düzenlemesi ve daha akıcı politika güncellemeleri sayesinde büyük giriş boyutlarına daha iyi tolerans gösterebileceğini düşündürmektedir. Ancak, bu geliştirilmiş MASAC varyantı dahi merkezi kredi dağıtımıyla hiç karşılaşmayan ISAC'dan daha düşük performans sergilemektedir. Bu durum, MASAC'nin büyük girişlere dayanıklılığına rağmen kredi dağıtımının hâlâ bir zorluk olduğunu göstermektedir. Özellikle, önceki deneylerde RRT*+MADDPG ve RRT*+MASAC'nin düşük performansı bu bulgularla uyumludur.

Birlikte, bu bulgular merkezi off-policy çok ajanlı pekiştirmeli öğrenme yöntemlerinin performansını hem eleştirmen girişinin büyüklüğünün hem de paylaşılan kredi dağıtımının zorluğunun düşürebileceğini doğrulamaktadır. Bu sonuçları Tablo 4'te sunmaktayız.

Karşılaştırma Ölçütü Haritaları

CAMAR, çok etmenli senaryolarda navigasyon ve koordinasyonun değerlendirilmesi için tasarlanmış üç farklı karşılaştırma ölçütü haritası içerir. Bu haritalar, düzen, karmaşıklık ve ajan konfigürasyonları açısından farklılık gösterir.

Rastgele ızgara haritası, çember tabanlı ayrıklaştırmaya sahip rastgele yerleştirilmiş dikdörtgen engeller içeren 20×20 ızgaradır. Ajanlar boş hücrelerde başlatılır ve hedefler rastgele seçilen konumlara yerleştirilir. Senaryo, holonomik dinamikler kullanır ve engel yoğunluğu, ajan yarıçapı, hedef yarıçapı ile diğer parametreleri içerir. Karşılaştırma sırasında, altı farklı rastgele ızgara örneği oluşturulur: 8 ajanlı üç harita ve 32 ajanlı üç harita. Bu varyasyonların her biri, haritanın karmaşıklığını belirleyen üç farklı engel yoğunluğu (0.0, 0.05, 0.15) altında değerlendirilir. Bu görevler nispeten basittir, ancak çok etmenli yol bulma (ÇEY) algoritmalarını test etmek için hala faydalıdır.

Labmaze ızgarası, prosedürel labirent oluşturulmasına dayanır . Birden çok dikdörtgen oda ve koridor kullanarak daha yapılandırılmış haritalar oluşturur. Ek bağlantı olasılığı parametresi odalar arasındaki bağlantı sayısını kontrol eder: 1.0 değeri , haritanın tamamen bağlı olduğu anlamına gelirken, daha düşük değerler daha fazla izole alan ve darboğazlar oluşturur . Rastgele ızgara ile aynı şekilde, 6 örnek değerlendirilir: 8 ajanlı üç harita ve 32 ajanlı üç harita, her biri üç farklı bağlantı olasılığı altında (0.4, 0.65, 1.0). Dinamikler rastgele ızgaradakilerle aynıdır, bu da düzenler arasında tutarlı karşılaştırma yapılmasını sağlar. Bu görevler koordinasyon ve navigasyon için daha zorlayıcı meydan okumalar sunar.

Bu karşılaştırma ölçütü haritaları ölçeklenebilir, simülasyonu hızlıdır ve yapı ile zorluk üzerinde esnek kontrol sağlar. Birlikte , MAPF için kapsamlı bir test ortamı oluştururlar.

Heterojen Destek

The `hetero give way` haritası heterojen çok ajanlı koordinasyonu test eder. Bu, daha büyük bir ajanın ortadaki alana giremediği basit bir koridor senaryosuna dayanır. daha küçük ajan ise geçiş yapabilir. Bu yapı, ajanların örtük olarak sıra alma veya öncelik tanıma davranışını öğrenmelerini zorunlu kılar. Dinamikler aynı kalmakla birlikte (`Holonomicdynamic`), bu yapı ajan yeteneklerindeki farklılıkları vurgular. Temel algoritmalar olan MAPPO ve IPPO bu görevde zorluk yaşamaktadır. Buna rağmen, CAMAR kutudan çıktığı haliyle böyle heterojen ayarları destekler ve farklı büyüklük, dinamik ve davranışa sahip ajanların koordinasyon stratejileri üzerine yapılan araştırmaları ilerletmek için kullanılabilir.

Ek E — Genişletilmiş Ölçeklenebilirlik Analizi

CAMAR'ın ölçeklenebilirliğini, sınırlarını daha iyi anlamak amacıyla popüler diğer MARL ortamları ile karşılaştırarak inceliyoruz.

Diğer Ortamlarla Karşılaştırma**.** Tablo 5'te, CAMAR'ı JaxMARL ve Jumanji gibi GPU destekli diğer ortamlarla karşılaştırıyoruz. CAMAR, VMAS ve MPE tabanlı ortamlardan daha hızlıdır ve en iyi GPU simülatörlerine yakın hızlara ulaşmaktadır. Örneğin, CAMAR 4 ajanla 338K SPS ve 32 ajanla 50K SPS hızında çalışmaktadır. Bazı Jax- MARL ortamları daha yüksek verimlilik sağlar—örneğin, SMAX 1.2M SPS'ye kadar ulaşır—ancak ajan sayısı arttıkça performansları daha çabuk düşer. Örneğin , JaxNav 32 ajan sayısından fazla ile çalışamaz. SMAX ve RWARE güçlü verimlilik sergilemekle birlikte, açık alan veya ayrık ızgara gibi basitleştirilmiş ortamları kullanır ve işlemsel olarak üretilen engel düzenlerini desteklemezler. Buna karşılık CAMAR sürekli kontrol ve yoğun engelleri destekleyerek daha yüksek gerçeklik düzeyi sunar. Gerçek dünya MAPF zorluklarıyla daha iyi uyum sağlayan daha zengin bir ortam için küçük bir hız kaybını kabul eder.

Ajanlar	4	8	16	32	64	128
CAMAR	338K	189K	99K	50K	25K	13K
VMAS	16K	10K	5,5K	2,7K 1,2K	447	
JaxNav	159K	61K	13,6K	353	-	-
SMAX	0.8M	1.0M	1.2M	0.8M	0.3M	0.1M
RWARE	1.4M	1.1M	0.8M	0.5M	0.3M	0.1M
MPE-2	0.6M	0.8M	1.2M	0.7M	0.5M	0.2M
MPE-1120	11.8K	11.2K	11.9K	11.1K	9.7K	7.7K

Tablo 5: SPS ile ajan sayısının karşılaştırılması. GPU destekli ortamlar arasında karşılaştırma. MPE-2, 2 çember engel içeren varsayılan simpletag yapılandırmasını kullanırken, MPE-1120 CAMAR'da bulunan çember engel sayısına karşılık gelecek şekilde 1120 çember kullanmaktadır. '-' işareti, ortamın bölümü tamamlayacak kadar hızlı olmadığını göstermektedir.

Ek F — Gelecek Çalışmalar

CAMAR kullanılarak araştırılabilecek çeşitli araştırma yönlerini vurgulamaktayız.

İlk olarak, CAMAR çok sayıda ajanla test yapılmasına olanak sağlar ve bu, ölçeklenebilir çok ajanlı takviyeli öğrenme (ÇATL) yöntemlerinin geliştirilmesini destekler. Gelecek çalışmalar, algoritmaların karmaşık ortamlarda yüzlerce ajanla başa çıkma yeteneklerini iyileştirmeye odaklanabilir.

İkinci olarak, CAMAR ajanlar arasındaki iletişimin incelenmesi için iyi bir platform sunar. Büyük ölçekli görevlerde ajanlar sıklıkla

need to share information with nearby teammates. This is an important challenge that is still not well studied. CAMAR provides the tools to explore this type of localized communication.

Third, CAMAR includes realistic navigation tasks, which makes it useful for combining planning methods (like RRT, RRT*) with learning-based agents. This combination could improve both sample efficiency and robustness by using long-term planning together with learned policies.

Fourth, CAMAR supports agents with different abilities and dynamics. This enables research on heterogeneous teams, where agents may have different sizes, speeds, or types of control. In future work, researchers can develop methods that automatically adapt to such diverse agent groups.

We hope CAMAR will support the community in studying these and other open problems in multi-agent learning and planning.

Appendix G — Implementation Details

Integrations

CAMAR is designed to work easily with modern reinforcement learning tools. It follows the Gymnax interface, which is already familiar to many researchers working with RL environments on top of JAX [30].

We also provide a wrapper for TorchRL [35], which allows users to integrate CAMAR into PyTorch-based pipelines. In addition, CAMAR supports integration with BenchMARL [36], a framework for evaluating MARL algorithms. These integrations make it easy to use CAMAR with popular RL frameworks.

Vectorized setup

To maintain high simulation speed, CAMAR uses efficient vectorized operations based on JAX [30]. Agents with the same dynamic model are grouped together and updated in parallel. For agent sizes, two cases are supported: if all agents have the same radius, it is treated as a constant during JAX compilation [30], giving faster simulation. If agents have different sizes, their radii are passed as vectors being part of the environment state, which still allows efficient processing and supports randomized sizes during training.

Appendix H — Extended Related Work

Waterworld (SISL) The Waterworld environment, part of the SISL (Stanford Intelligent Systems Laboratory) suite [24], is a continuous control benchmark where multiple agents move in a bounded two-dimensional space to collect moving targets (“food”) while avoiding harmful objects (“poison”). Agents have continuous observations and actions, and their motion dynamics are simple and lightweight, which allows high simulation speeds for training.

The environment places a small number of obstacles randomly in the scene, with the default setup containing only one obstacle. This limited variation does not require agents to generalize across different maps in a meaningful way. Waterworld does not support heterogeneous agents, evaluation protocols, or complex multi-stage tasks, and the scenario remains structurally the same across runs.

Multi-Robot Warehouse (RWare) The Multi-Robot Warehouse environment [11] simulates robots moving in a warehouse to collect and deliver requested goods. The layout is grid-based, and agents must coordinate to navigate around shelves and other robots. The default version is implemented in Python using a discrete state and action space. It supports partial observability but does not provide procedural generation: the warehouse layout is fixed. As a result, agents do not need to generalize across different maps.

Several re-implementations of RWare exist. The Jumanji version [19] rewrites the environment in JAX, which improves simulation speed and allows hardware acceleration on GPUs, but keeps the original fixed-layout design and task structure. The PufferLib version [25] modifies the environment to support large-scale parallel simulation.

None of the RWare versions support heterogeneous agents, continuous control, or evaluation protocols. Despite this, RWare remains a widely used benchmark for discrete-space multi-agent pathfinding and cooperative delivery tasks.

Trash Pickup (PufferLib) The Trash Pickup environment [25] is a grid-based multi-agent task where agents move around a map to collect pieces of trash and deliver them to designated drop-off locations. It is implemented in C with a Python API for controlling agents, which allows efficient simulation. The environment uses discrete state and action spaces and supports partial observability.

The layout is fixed, and trash positions follow a predefined spawn pattern. While trash locations may vary between episodes, the underlying map structure remains the same. This means the environment does not require generalization across different maps. The design supports scalability to more than 500 agents.

The Trash Pickup benchmark does not provide continuous control, heterogeneous agents, or formal evaluation protocols. However, it offers a simple and repeatable cooperative task that can be scaled to large numbers of agents, making it a useful test case for studying coordination efficiency in discrete environments.

StarCraft Multi-Agent Challenge (SMAC) The StarCraft Multi-Agent Challenge [18] is one of the most widely used benchmarks in the MARL community. In SMAC, a team of StarCraft II units controlled by independent agents must cooperate to defeat an opposing team. The environment is partially observable, and by default uses discrete action spaces. The underlying maps are fixed, which allows agents to solve the tasks without relying on observation inputs by simply memorizing the optimal action sequence for each map. This significantly reduces the need for generalization across scenarios.

SMACv2 SMACv2 [15] addresses one of the key limitations of SMAC by introducing randomly generated maps with random positions of units, which prevents memorization of fixed action sequences and forces agents to generalize their policies across start positions. The rest of the environment remains the same as SMAC, with partially observable states and discrete actions by default. Like SMAC, SMACv2 does not include an evaluation protocol, but it can be evaluated

yakınlardaki takım arkadaşlarıyla bilgi paylaşmak zorundadır . Bu, hâlâ yeterince incelenmemiş önemli bir zorluktur. CAMAR, bu tür yerel iletişimi araştırmak için gerekli araçları sağlar.

Üçüncü olarak, CAMAR gerçekçi navigasyon görevlerini içerir; bu da planlama yöntemleri (RRT, RRT* gibi) ile öğrenmeye dayalı ajanların entegrasyonu için faydalıdır. Bu entegrasyon, uzun vadeli planlamayı öğrenilmiş politikalarla birlikte kullanarak örnek verimliliği ve sağlamlığı artırabilir.

Dördüncü olarak, CAMAR farklı yetenek ve dinamiklere sahip ajanları destekler. Bu, ajanların farklı boyutlara, hızlara veya kontrol türlerine sahip olabileceği heterojen takımlar üzerine araştırma yapılmasını mümkün kılar. Gelecek çalışmalarda, araştırmacılar bu tür çeşitli ajan gruplarına otomatik uyum sağlayabilen yöntemler geliştirebilir.

CAMAR’ın, çok etmenli öğrenme ve planlama alanlarındaki bu ve diğer açık problemler üzerinde topluluğa destek sağlamasını umuyoruz.

Ek G — Uygulama Detayları

Entegrasyonlar

CAMAR, modern takviyeli öğrenme araçlarıyla sorunsuz çalışacak şekilde tasarlanmıştır. JAX üzerinde çalışan RL ortamlarıyla çalışan birçok araştırmacıya aşina olan Gymnax arayüzünü takip eder [30].

Ayrıca, kullanıcıların CAMAR’ı PyTorch tabanlı süreçlere entegre etmelerini sağlayan TorchRL [35] için bir sarmalayıcı sunar.
Buna ek olarak, CAMAR, MARL algoritmalarını değerlendirmek için kullanılan BenchMARL [36] entegrasyonunu desteklemektedir.
Bu entegrasyonlar, CAMAR’ın popüler RL çerçeveleriyle kullanımını kolaylaştırır.

Vektörleştirilmiş yapılandırma

Yüksek simülasyon hızını korumak için CAMAR, JAX [30] tabanlı verimli vektörleştirilmiş işlemler kullanır. Aynı dinamik modele sahip ajanlar birlikte gruplanır ve paralel olarak güncellenir.
Ajan boyutları için iki durum desteklenmektedir: tüm ajanların yarıçapı aynıysa, bu JAX derlemesi sırasında sabit olarak kabul edilir [30] ve böylece daha hızlı simülasyon sağlanır. Ajanların farklı boyutları varsa, yarıçapları çevre durumunun bir parçası olan vektörler olarak iletilir; bu, hâlâ verimli işlemeye olanak tanır ve eğitim sırasında rastgele boyut desteği sağlar.

Ek H — Genişletilmiş İlgili Çalışmalar

Waterworld (SISL) Waterworld ortamı, SISL (Stanford Akıllı Sistemler Laboratuvarı) paketinin bir parçası olup [24], çoklu ajanların sınırlandırılmış iki boyutlu bir alanda hareket ederek hareketli hedefleri (“gıda”) topladığı ve zararlı nesnelerden (“zehir”) kaçındığı sürekli kontrol karşılaştırma ölçütüdür. Ajanlar sürekli gözlem ve eylemlere sahiptir ve hareket dinamikleri basit ve hafiftir; bu da eğitim için yüksek simülasyon hızlarına imkân tanır.

Ortamda sahneye rastgele az sayıda engel yerleştirilir; varsayılan ayarda yalnızca bir engel bulunmaktadır. Bu sınırlı varyasyon, ajanların farklı haritalar arasında anlamlı bir şekilde genelleme yapmasını gerektirmez. Waterworld, heterojen ajanları, değerlendirme protokollerini veya karmaşık çok aşamalı görevleri desteklemez ve senaryo çalıştırmalar arasında yapısal olarak aynıdır.

Çok Robotlu Depo (RWare) Çok Robotlu Depo ortamı [11], robotların depoda istenen malları toplaması ve teslim etmesini simüle eder. Yerleşim ızgara tabanlıdır ve ajanların rafların ve diğer robotların etrafında gezinmek için koordinasyon sağlaması gerekir. Varsayılan sürüm, ayrık durum ve eylem uzayı kullanılarak Python’da uygulanmıştır. Kısmi gözlemlenebilirliği destekler ancak prosedürel üretim sağlamaz: depo yerleşimi sabittir. Sonuç olarak, ajanların farklı haritalar arasında genelleme yapması gerekmez.

RWare için çeşitli yeniden uygulamalar mevcuttur. Jumanji sürümü [19], ortamı JAX ile yeniden yazarak simülasyon hızını artırır ve GPU’larda donanım hızlandırmasına olanak tanır, ancak orijinal sabit yerleşim tasarımı ve görev yapısını korur. PufferLib versiyonu [25], büyük ölçekli paralel simülasyonu desteklemek için ortamı değiştirmektedir.

RWare versiyonlarının hiçbiri heterojen ajanları, sürekli kontrolü veya değerlendirme protokollerini desteklememektedir. Buna rağmen, RWare ayrık uzaylı çok etmenli yol bulma ve iş birliğine dayalı teslimat görevleri için yaygın olarak kullanılan bir karşılaştırma ölçütü olmaya devam etmektedir.

Çöp Toplama (PufferLib) Çöp Toplama ortamı [25], ajanların harita üzerinde hareket ederek çöp parçalarını topladığı ve belirlenmiş bırakma noktalarına teslim ettiği ızgara tabanlı çok etmenli bir görevdir. C dilinde uygulanmış olup, ajanların kontrolü için Python API’si sunmakta ve böylece verimli simülasyona imkan sağlamaktadır. Ortam, ayrık durum ve eylem uzayları kullanmakta ve kısmi gözlemlenebilirliği desteklemektedir.

Yerleşim düzeni sabit olup, çöp konumları önceden tanımlanmış bir doğma modelini takip etmektedir. Çöp konumları bölüm aralarında değişiklik gösterebilmekle birlikte temel harita yapısı aynıdır.
Bu, ortamın farklı haritalar arasında genelleme gerektirmediği anlamına gelmektedir. Tasarım, 500’ün üzerinde ajan sayısına ölçeklenebilirliği desteklemektedir.
Çöp Toplama karşılaştırma ölçütü sürekli kontrol, heterojen ajanlar veya resmi değerlendirme protokolleri sağlamamaktadır.
Bununla birlikte, çok sayıda ajanla ölçeklendirilebilen basit ve tekrarlanabilir bir işbirlikçi görev sunar; bu da ayrık ortamlarındaki koordinasyon verimliliğinin incelenmesi için faydalı bir test durumu oluşturur.

StarCraft Çok Ajanlı Mücadele (SMAC) StarCraft Çok Ajanlı Mücadele [18] , Çok Ajanlı Takviyeli Öğrenme (MARL) topluluğunda en yaygın kullanılan karşılaştırma ölçütlerinden biridir. SMAC'ta, bağımsız ajanlar tarafından kontrol edilen bir StarCraft II birimleri takımı, karşıt bir takımı mağlup etmek için işbirliği yapmak zorundadır. Ortam kısmen gözlemlenebilir olup, varsayılan olarak ayrık eylem uzaylarını kullanmaktadır.
Alt yapıyı oluşturan haritalar sabittir; bu durum, ajanların her harita için optimal eylem dizisini ezberleyerek görevleri gözlem girdilerine bağımlı olmadan çözmelerine olanak tanır. Bu durum, senaryolar arası genelleştirme gereksinimini önemli ölçüde azaltır.

SMACv2 SMACv2 [15], sabit eylem dizilerinin ezberlenmesini engelleyen ve ajanların politika genellemesini başlangıç pozisyonları arasında zorunlu kılan rastgele oluşturulmuş haritalar ve birimlerin rastgele konumları ile SMAC'in temel sınırlamalarından birini ele almaktadır. Çevrenin diğer kısımları, varsayılan olarak kısmi gözlemlenebilir durumlar ve ayrık eylemlerle SMAC ile aynıdır. SMAC gibi, SMACv2 de bir değerlendirme protokolü içermez; ancak

using the protocol proposed in [32].

SMAX (JaxMARL) SMAX [17] is a JAX-based re-implementation of SMACv2 that leverages hardware acceleration for faster simulation. In addition to performance improvements, SMAX introduces the option to switch from discrete to continuous action spaces, making it more flexible for testing different types of MARL algorithms.

Multi-Agent Particle Environment (MPE) The Multi-Agent Particle Environment [20, 16] is a lightweight 2D simulator for testing cooperative, competitive, and communication multi-agent tasks. Agents and landmarks are represented as simple geometric shapes, and their interactions follow basic physical dynamics. MPE supports both continuous and discrete action spaces, and scenarios can be either fully or partially observable. Procedural generation is not used in the default scenarios, although obstacle or landmark positions can be randomized in some tasks. This means generalization across different maps is not strictly required. The environment does not include a built-in evaluation protocol.

MPE (JaxMARL) The JaxMARL re-implementation of MPE [17] offers hardware acceleration through JAX, which enables much faster simulation compared to the original Python implementation. Functionally, it mirrors MPE in terms of available scenarios, physics, and agent capabilities. It supports both continuous and discrete action spaces. Like the original MPE, it does not include procedural map generation by default and does not provide an evaluation protocol.

JaxNav (JaxMARL) JaxNav [26] is a navigation-focused benchmark implemented within the JaxMARL framework. It places agents in a continuous 2D space where they must navigate to goal locations while avoiding static obstacles. The environment supports both continuous and discrete action spaces. Maps are fixed in the default setup, so agents do not need to generalize to unseen layouts. JaxNav is implemented in JAX for hardware acceleration but does not provide an evaluation protocol.

Nocturne Nocturne [27] is a 2D partially observed driving simulator implemented in C++ for performance, with a Python API for training and evaluation. It focuses on realistic autonomous driving scenarios sourced from the Waymo Open Dataset. The environment provides a set of benchmark tasks in which agents control autonomous vehicles interacting with traffic participants whose trajectories are taken from recorded data. This setup captures complex multi-agent interactions requiring coordination, prediction of other agents' intentions, and handling of partial observability.

Nocturne does not use procedurally generated maps, as all scenes are drawn from fixed datasets. While the simulator benefits from its C++ implementation, it does not provide GPU acceleration, which limits its throughput for large-scale MARL experiments.

POGEMA The Partially Observable Grid Environment for Multiple Agents [12] is a grid-based benchmark for partially observable multi-agent pathfinding. Agents operate with ego-centric local observations and must reach individual goals while avoiding collisions. A key feature of POGEMA is its procedural map generation, producing diverse layouts—random maps, mazes, and warehouse-style scenarios—that require agents to generalize to unseen environments. It also offers integrated evaluation protocols with standardized metrics, enabling consistent comparison across reinforcement learning, planning, and hybrid methods. Implemented in Python, POGEMA supports scalable multi-agent experiments with both classical and learning-based policies.

VMAS The Vectorized Multi-Agent Simulator for Collective Robot Learning [14] is a PyTorch-based, vectorized 2D physics framework designed for efficient multi-robot benchmarking. It includes a modular interface for defining custom scenarios, alongside a set of built-in multi-robot tasks, that each involve a relatively small number of agents. VMAS stands out for its GPU-accelerated, batch-simulated environments. It supports inter-agent communication and sensors (e.g., LIDAR).

SMART Scalable Multi-Agent Realistic Testbed [28] is a physics-based simulator tailored for bridging Multi-Agent Path Finding (MAPF) algorithms and real-world performance. Designed for scalability, SMART supports simulation of thousands of agents, enabling evaluation of large-scale deployments. However, its simulation speed is relatively low (around 335 SPS for 100 agents), which can limit large-scale reinforcement learning experiments. The platform targets both academic researchers and industrial users who lack access to extensive physical robot fleets, offering a realistic environment to test MAPF performance under near-real conditions.

Gazebo Gazebo is a widely used open-source 3D robotics simulator that integrates tightly with ROS. It supports detailed physics simulation, realistic sensors (e.g., LIDAR, cameras), and high-quality rendering. While Gazebo allows multi-robot experimentation and moderate scalability, it does not support GPU acceleration for simulation logic - GPU use is limited to rendering and sensor visuals only. This limitation can make simulations of large multi-agent systems slow or resource-heavy.

Webots Webots is a commercial-grade 3D robotics simulator offering a wide array of built-in robot models, sensors, and actuator libraries. It provides realistic physics simulation, high-quality graphics, and ROS support. Webots is well-suited for prototyping and academic or industrial robotic research. However, from a multi-agent perspective, it's not optimized for running large numbers of agents at scale—simulation speed tends to drop significantly as agent count grows, making massive multi-robot evaluations challenging.

ARGoS ARGoS [23] is an open-source, modular simulator designed specifically for swarm robotics and large-scale multi-agent systems. Its architecture supports running thousands of simple agents efficiently by combining multiple physics engines and leveraging parallel computation. ARGoS is customizable, allowing researchers to define their own robots, sensors, actuators, etc.

[32]'de önerilen protokol kullanılarak değerlendirilebilir.

SMAX (JaxMARL) SMAX [17], daha hızlı simülasyon için donanım hızlandırmasından yararlanan JAX tabanlı bir SMACv2 yeniden uygulamasıdır. Performans iyileştirmelerine ek olarak SMAX, ayrık eylem alanlarından sürekli eylem alanlarına geçiş seçeneği sunarak farklı türdeki MARL algoritmalarının test edilmesini daha esnek hale getirir.

Çok Eylemli Parçacık Ortamı (MPE) Çok Eylemli Parçacık Ortamı [20, 16], işbirlikçi, rekabetçi ve iletişim gerektiren çok ajanlı görevlerin test edilmesi için hafif bir 2B simülatördür. Ajanlar ve işaret noktaları basit geometrik şekillerle temsil edilir ve etkileşimleri temel fiziksel dinamiklere dayanır. MPE hem sürekli hem de ayrık eylem alanlarını destekler ve senaryolar tamamen ya da kısmen gözlemlenebilir olabilir. Varsayılan senaryolarda prosedürel üretim kullanılmaz , ancak bazı görevlerde engel veya işaret konumları rastgeleleştirilebilir. Bu, farklı haritalar arasında genellemenin kesin olarak zorunlu olmadığı anlamına gelir. Ortamda yerleşik bir değerlendirme protokolü bulunmamaktadır.

MPE (JaxMARL) MPE'nin [17] JaxMARL yeniden uygulaması , JAX aracılığıyla donanım hızlandırması sunar; bu da orijinal Python uygulamasına kıyasla çok daha hızlı simülasyon sağlar. Fonksiyonel olarak, mevcut senaryolar, fizik ve ajan yetenekleri açısından MPE ile aynıdır. Hem sürekli hem de ayrık eylem alanlarını destekler.
Orijinal MPE gibi, varsayılan olarak prosedürel harita üretimi içermez ve bir değerlendirme protokolü sağlamaz.

JaxNav (JaxMARL) JaxNav [26], JaxMARL çerçevesinde uygulanmış navigasyona odaklı bir karşılaştırma ölçütüdür. Ajanlar, statik engellerden kaçınarak hedef konumlara yönelmeleri gereken sürekli 2B bir alanda konumlandırılır. Ortam, hem sürekli hem de ayrık eylem alanlarını desteklemektedir. Haritalar varsayılan yapılandırmada sabittir; bu nedenle ajanların daha önce görülmemiş düzenlere genelleme yapmasına gerek yoktur. JaxNav, donanım hızlandırması için JAX ile uygulanmıştır ancak bir değerlendirme protokolü sağlamamaktadır.

Nocturne Nocturne [27], performans için C++ ile uygulanmış ve eğitim ile değerlendirme için Python API'si bulunan 2B kısmi gözlemlenebilir sürüş simülatörüdür. Gerçekçi otonom sürüş senaryolarına, Waymo Open Dataset'ten sağlanan verilere odaklanmaktadır. Ortam, ajanların trafik katılımcılarının kayıtlı veri yol izleriyle etkileşime giren otonom araçları kontrol ettiği bir dizi karşılaştırma ölçütü görevi sunmaktadır. Bu yapılandırma, koordinasyon gerektiren karmaşık çok ajanlı etkileşimleri, diğer ajanların niyetlerinin tahmin edilmesini ve kısmi gözlemlenebilirlik durumunun yönetilmesini kapsamaktadır.

Nocturne, tüm sahneler sabit veri setlerinden alındığı için prosedürel olarak oluşturulmuş haritalar kullanmamaktadır. Simülatör C++ uygulamasından yararlansa da GPU hızlandırması sağlamamaktadır; bu da büyük ölçekli Çok Ajanlı Takviyeli Öğrenme deneylerinde verimini sınırlandırmaktadır.

POGEMA Çok Ajanlar için Kısmen Gözlemlenebilir Grid Ortamı [12], kısmen gözlemlenebilir çok etmenli yol bulma için grid tabanlı bir karşılaştırma ölçütüdür. Ajanlar, ego-merkezli yerel gözlemlerle hareket eder ve bireysel hedeflere ulaşmak zorundadırlar, aynı zamanda çarpışmalardan kaçınmalıdırlar. POGEMA'nın temel özelliklerinden biri, ajanların görülmemiş ortamlara genelleme yapmasını gerektiren çeşitli düzenler üreten prosedürel harita üretimidir—rastgele haritalar, labirentler ve depo tarzı senaryolar. Ayrıca, takviyeli öğrenme, planlama ve karma yöntemler arasında tutarlı karşılaştırmaya olanak tanıyan standartlaştırılmış metriklerle entegre değerlendirme protokolleri sunar. Python ile uygulanmış olan POGEMA, hem klasik hem de öğrenmeye dayalı politikalarla ölçeklenebilir çok ajanlı deneyleri desteklemektedir.

VMAS Kolektif Robot Öğrenmesi için Vektörize Çok Ajanlı Simülatör [14], verimli çok robotlu karşılaştırmalar için PyTorch tabanlı, vektörize 2B fizik çerçevesidir. Özelleştirilmiş senaryolar tanımlamak için modüler bir arayüz ile her biri nispeten az sayıda ajan içeren yerleşik çok robotlu görevler setini içerir. VMAS, GPU hızlandırmalı, toplu simülasyonlu ortamları ile öne çıkar. Ajanlar arası iletişim ve sensörler (örneğin, LIDAR) desteklemektedir.

SMART Ölçeklenebilir Çok Ajanlı Gerçekçi Test Ortamı [28] , Çok Etmenli Yol Bulma (MAPF) algoritmaları ile gerçek dünyadaki performansı birleştirmeye yönelik fizik tabanlı bir simülatördür. Ölçeklenebilirlik üzere tasarlanan SMART , binlerce ajanın simülasyonunu destekleyerek büyük ölçekli dağıtımların değerlendirilmesini mümkün kılar. Ancak , simülasyon hızı nispeten düşüktür (100 ajan için yaklaşık 3 35 SPS), bu durum büyük ölçekli takviyeli öğrenme deneylerini sınırlandırabilir. Platform, geniş fiziksel robot filolarına erişimi olmayan akademik araştırmacı ve endüstri kullanıcılarını hedefleyerek MAPF performansını gerçek koşullara yakın bir ortamda test etme imkânı sunar.

Gazebo Gazebo, ROS ile sıkı entegrasyona sahip, yaygın kullanılan açık kaynak kodlu 3D robotik simülatördür. Ayrıntılı fizik simülasyonu, gerçekçi sensörler (örneğin, LIDAR, kameralar) ve yüksek kaliteli görselleştirme imkânı sağlar. Gazebo, çoklu robot deneylerine ve orta düzey ölçeklenebilirliğe imkân tanırken, simülasyon mantığı için GPU hızlandırmasını desteklemez — GPU kullanımı yalnızca render ve sensör görselleştirmeleri ile sınırlıdır. Bu kısıtlama, büyük çok ajanlı sistemlerin simülasyonlarını yavaş veya kaynak açısından maliyetli hale getirebilir.

Webots Webots, geniş bir yerleşik robot modeli, sensör ve aktüatör kütüphaneleri yelpazesi sunan ticari kalitede 3D robot simülatörüdür. Gerçekçi fizik simülasyonu, yüksek kaliteli grafikler ve ROS entegrasyonu sağlar. Webots, prototipleme ile akademik veya endüstriyel robotik araştırmalar için oldukça uygundur. Ancak, çoklu ajan perspektifinden bakıldığında, çok sayıda ajanı ölçeklenebilir şekilde çalıştırmak için optimize edilmemiştir — ajan sayısı arttıkça simülasyon hızı önemli ölçüde düşer, bu da büyük çoklu robot değerlendirmelerini zorlaştırır.

ARGoS ARGoS [23], özellikle sürü robotik ve büyük ölçekli çok ajanlı sistemler için geliştirilmiş açık kaynaklı, modüler bir simülatördür. Mimari, birden fazla fizik motorunun birleşimi ve paralel hesaplamanın kullanımıyla binlerce basit ajanın verimli şekilde çalışmasını desteklemektedir. AR-GoS, araştırmacıların kendi robotlarını, sensörlerini, aktuatörlerini vb. tanımlamalarına imkân tanıyan özelleştirilebilir bir yapıya sahiptir.