Standardizing Benchmark Environments for Multi-Agent RL and a centralized approach towards MARL

Aman Chulawala, Vineet Tambe, Yulun Zang
Robotics Institute, Carnegie Mellon University
achulawa@andrew.cmu.edu, vtambe@andrew.cmu.edu, yulunz@andrew.cmu.edu

Abstract: We implement a centralized Multi-Agent Reinforcement Learning (MARL) environment for Multi-Agent Path Finding (MAPF) algorithms. Previous works on MARL lack agreement on critical design choices such as information sharing, action space, and edge weights. We implement a new MARL environment based on gymnasium, unifying the MARL formulation. We then train a simple centralized MARL policy to demonstrate the applicability of the environment.

Keywords: Multi-Agent Path Finding, Multi-Agent Reinforcement Learning

1 Introduction

We study the problem of implementing a centralized Multi-Agent Reinforcement Learning (MARL) environment for Multi-Agent Path Finding (MAPF) problems. MAPF is the problem of looking for collision-free paths for a group of agents from their corresponding start to goal locations. The importance of MAPF comes from its wide application in multi-robot coordination systems in automated warehouses, multi-robot assembly in manufacturing scenarios, and traffic coordination in crossovers. The designed environment can be found at here and the multi-agent -rl policy and training code can be found at here.

2 Prior Work

Previous works on Multi-Agent Path Finding (MAPF) [1, 2, 3, 4, 5, 6, 7, 8, 9] are broadly divided into centralized and decentralized methods. Centralized methods involve a planner with complete MAPF information, generating collision-free paths for each agent. Decentralized methods, lacking such a planner, rely on agents' individual, partial observations for path computation. Centralized methods typically perform better due to more information, but face practical challenges like communication issues. Decentralized methods, while more practical and robust to real-world variables, often yield inferior solutions due to limited observability.

Meanwhile, Multi-Agent Reinforcement Learning (MARL) methods [10, 11] lie in between centralized and decentralized methods. MARL methods are decentralized by nature because they usually train a policy for each agent. However, the MARL framework can be adapted to include the sharing of information within the agents' observation space. By adjusting the degree of information sharing, MARL methods can flexibly transition across the spectrum of centralized and decentralized methods. Since state-of-the-art MARL methods [10, 11] are more decentralized than centralized, in this project, we would like to explore a fully centralized MARL formulation because of the advantage of fully shared information among agents.

On the other hand, previous MARL methods [10, 11] lack agreement on other problem formulations such as action space and edge weights. For action space, some make the agents move in a 4-neighbor

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

grid in an omnidirectional fashion (i.e. left, right, up, down), while others have the agents rotate left or right for 90 degrees and move forward. The former is used more extensively in MAPF research, and the latter is more realistic in the real world. For edge weights, to the authors' best knowledge, no previous works have explicitly considered edge weights as part of the observation. Therefore, we unify the MARL problem formulation by implementing a new multi-agent environment based on openai-gym API.

We make the following contributions: (1) we implement a new multi-agent environment based on openai-gym API that considers the more realistic agent action space and takes edge weights into consideration, (2) we train a centralized MARL shared policy in the implemented environment to demonstrate the applicability of the implemented environment.

3 Problem Formulation

Multi-Agent Path Finding (MAPF) attempts to find collision-free paths for a group of agents from their corresponding start to goal locations. Formally, given in a graph $G(V, E)$, n agents a_i, and their corresponding start and goal locations $s_i \in V$, $g_i \in V$, where $0 \leq i < n$, MAPF attempts to find a path π_i for each agent such that the sum of path lengths $\sum_{i=0}^{n-1} |\pi_i|$ is minimized and the paths have no collisions. We consider two possible kinds of collisions, namely vertex conflicts and edge conflicts. Vertex conflicts happen when two agents attempt to go to the same vertex at the same timestep. Edge conflicts happen when two adjacent agents attempt to swap locations at the same timestep.

4 Approach

4.1 Envrinment Modeling

The MARL and MAPF communities are trying to solve the same problem with different approaches. However, the lack of a common environment prevents an apples-to-apples comparison of the results. To resolve this we built a gymnasium-minigrid based simulation environment that can load map layouts from the standard benchmark mapf configuration or scenario files. A few examples of the environment with various configuration files can be seen in the figures 1 and 2.

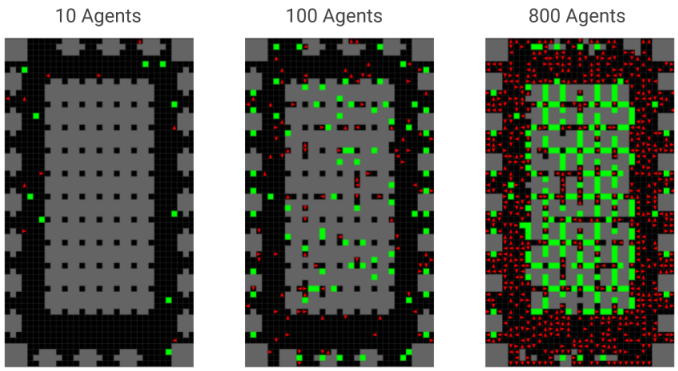

Figure 1: A visualization of the environment simulating 10, 100, and 800 agents respectively in the standard mapf warehouse map.

4.1.1 Observation Space

The observation space is a list of 7xHxW for N agents i.e. Nx7xHxW for N agents. The H and W parameters can be passed as environment arguments and represent the height and width of the view that the agent see's in front of it. Every channel in the HxW image represents the following:

Figure 2: A visualization of the environment simulating a city map with 100 agents.

1. Channel 1: A normalized egocentric view in front of the agent. (Top left of the figure 3.)
2. Channel 2: The goal location inside the agent's view. (Center left of the figure 3.)
3. Channel 3: Orientation of the agent. (Top right of the figure 3.)
4. Channel 4 to 7: Represent the edge weights in the 4 directions respectively as can be seen in the figure 3

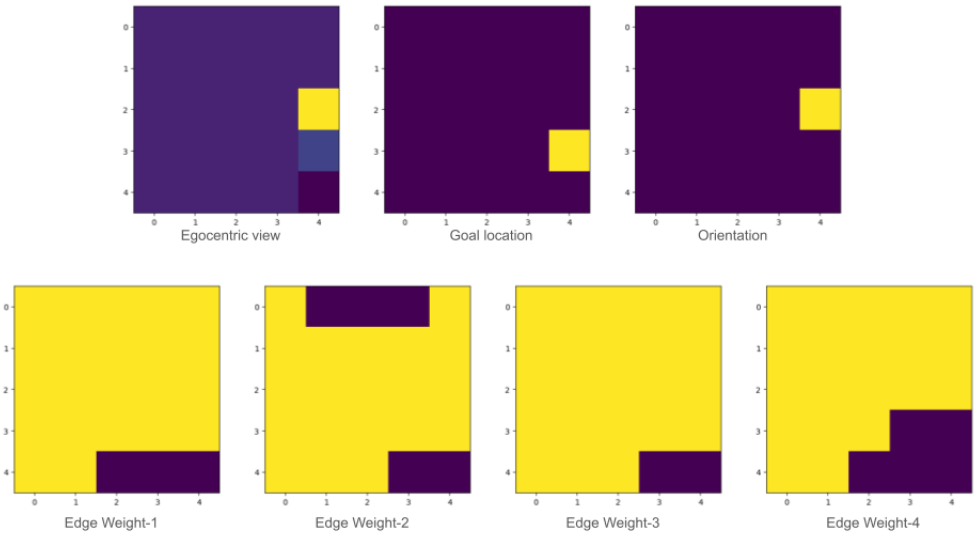

Figure 3: A visualization of the observation space for one agent. The H and W are set to 5 here.

4.1.2 Action Space

The agent is modeled as a point mass that can take the following 4 actions:

1. Forwards
2. Rotate Left
3. Rotate Right
4. Stay in place

The above-described action space is represented as a continuous-valued 1×4 vector representing a probability distribution over each of the above actions, and at runtime, the environment chooses an action with the maximum probability. For N agents, this will be represented as an $N \times 4$ matrix where every row represents the probability of actions for an agent.

To aid the training, we do an additional augmentation to the above action space:

We represent every action probability as a continuous value between $[-1, 1]$ which is then renormalized to $[0, 1]$, and resample the actions from the renormalized multinomial probability distribution. This can be expressed mathematically as follows:

$$\begin{aligned}
&\text{Action Space dimension} \quad A \in \mathbb{R}^{N \times 4}, N = \text{number of agents} \\
&\text{Augmented Action Space:} \quad A'_{ij} = \frac{\sigma(A_{ij}) + 1}{2} \quad \text{where } \sigma \text{ is the sigmoid function} \\
&\text{Resampled Action:} \quad \hat{A}_i \sim \text{Multinomial}(A'_{i \times 4})
\end{aligned}$$

4.1.3 Reward Function

We use a simple reward policy as follows:

$$Reward(s, a, s') = \begin{cases} 0.1 & \text{if } a = \text{forward} \\ 0.01 & \text{if } a = \text{rotate left} \\ 0.01 & \text{if } a = \text{rotate right} \\ 0 & \text{if } a = \text{stay in place} \\ 20 & \text{if goal is reached} \end{cases}$$

4.2 Termination

The episode terminates when all the agents have reached their goal location or the maximum number of time steps has been reached. The max-steps again is a configurable parameter that can be passed as an argument when creating the environment.

4.3 Policy Architecture & Training

In our work, we tackle this problem using a centralized approach. All the work that is out there approaches this problem by training decentralized policy for each agent or decentralized policy with "communication" [12] for MARL.

To do so we first stack the $N \times C \times H \times W$ observation (where C is the number of channels, $H \& W$ are the height and width of the observation images respectively) into a $N * C \times H \times W$ matrix - this is our state s. This is passed through a CNN feature extractor head to get a feature vector of size 128.

The CNN feature extractor has a simple architecture with 2 convolution layers and the output of this is passed to a single-layer MLP which gives a feature vector of size 128 as seen in Figure 4.

We use stable-baselines-3 to train our policy by leveraging the standardized implementation of PPO. The high-level architecture can be seen in Figure 5. The fully-connected network used for the policy and value function is a 2-layer MLP of size 64.

The trained model predicts a $N \times 4$ matrix which is our action matrix A given the stacked observation matrix s. This is passed to the environment via the 'step' function to get the next observation s'.

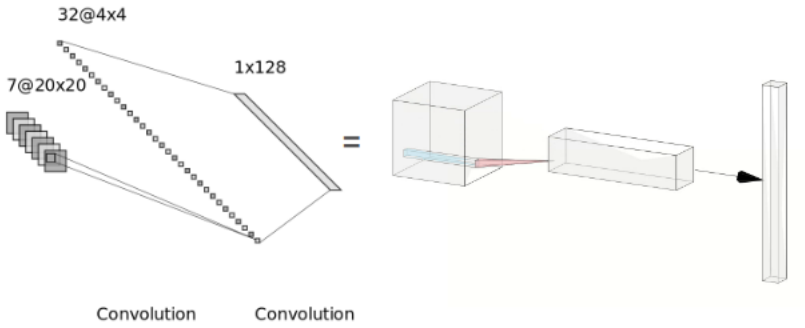

Figure 4: Architecture of the CNN feature extractor for input of size $7 \times 20 \times 20$

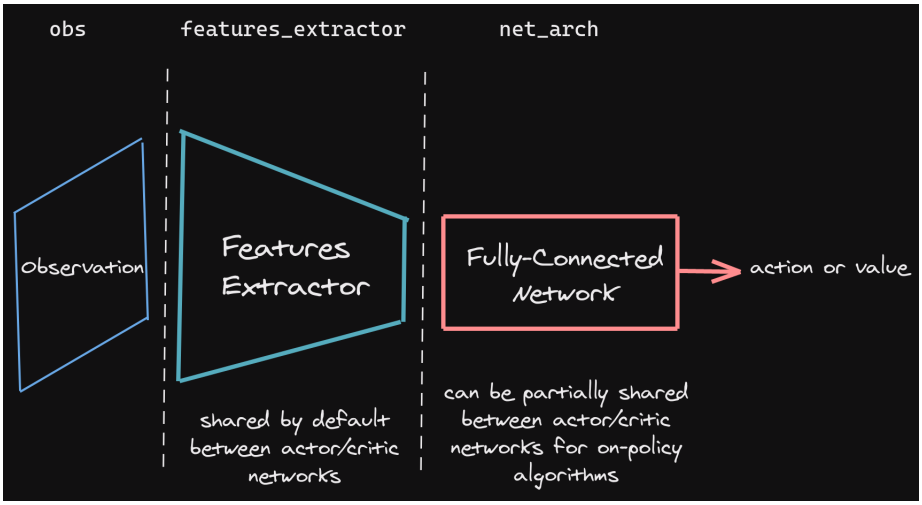

Figure 5: Policy and Value function Network Architecture (credits: stable-baselines-3)

5 Results & Summary

Through our work, we managed to accomplish the following:

1. A efficient simulation environment to train RL against where maps can be generated from benchmark mapf scenario configs.
2. A rich image observation space that comprises of
   (a) Egocentric view of the agent
   (b) goal position
   (c) orientation of the agent
   (d) edge weights
3. A continuous action space - capturing the probability of actions to take for each agent.
4. We were able to set up training of a centralized policy using PPO to predict the actions for each agent given the image observation for each agent, however, the policy performs poorly and is not able to route the agents to their goal location. We discuss ideas to improve the results in the Future Work Section. Figure 6 shows the various experiments we tried while training the model and improving the environment and our approach.

Figure 6: A screenshot of the mean eval reward curve for the various experiments tried during this project trying to figure out the correct hyperparameters to train the policy whilst improving the environment and our approach.

6 Conclusion and Future Work

In summary, our work introduces a centralized Multi-Agent Reinforcement Learning (MARL) environment for Multi-Agent Path Finding (MAPF). Built on the gymnasium-minigrid framework, our environment provides a standardized platform for benchmarking and collaborative research in the MARL community. Notable features include an efficient simulation setup for generating MAPF scenarios and a rich observation space that captures essential details like egocentric views, goal positions, agent orientations, and edge weights. Additionally, we employ a continuous action space, allowing agents to make probabilistic decisions.

While our initial attempts to train a centralized policy using Proximal Policy Optimization (PPO) revealed challenges in guiding agents effectively, this serves as a starting point for future improvements.

Looking forward, our focus lies on refining our MARL environment and training methodologies.

Key areas of exploration include:

1. Incorporating optimized edge weights in the observation to enhance decision-making.
2. Using normalized backward Dijkstra's path from the goal in the observation to aid the movement of the agent towards the goal.
3. Adding goal and agent current location directly to the feature vector of the CNN.
4. Training using the ideas of "Curiosity" as described in [13, 14].
5. Providing past actions and states as input to the model to improve generalization.
6. Given that we employee an image observation it is worth experimenting with the network architecture to get more dense feature vectors.

Acknowledgments

We would like to thank Rishi Veerapaneni for discussing and giving his thoughts on the approach and helping us with the initial concept.

References

[1] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, and W. Yeoh. Generalized target assignment and path finding using answer set programming. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1216–1223, 2017.

[2] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 837–845, 2017.

[3] M. Liu, H. Ma, J. Li, and S. Koenig. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1152–1160, 2019.

[4] J. Li, K. Sun, H. Ma, A. Felner, T. K. S. Kumar, and S. Koenig. Moving agents in formation in congested environments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 726–734, 2020.

[5] N. M. Kou, C. Peng, H. Ma, T. K. S. Kumar, and S. Koenig. Idle time optimization for target assignment and path finding in sortation centers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 9925–9932, 2020.

[6] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robotics and Automation Letters*, 6(3):5816–5823, 2021.

[7] A. Contini and A. Farinelli. Coordination approaches for multi-item pickup and delivery in logistic scenarios. *Robotics and Autonomous Systems*, 146:103871, 2021. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2021.103871. URL https://www.sciencedirect.com/science/article/pii/S0921889021001561.

[8] S. Varambally, J. Li, and S. Koenig. Which MAPF model works best for automated warehousing? In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 190–198, 2022.

[9] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning - lifelong. *IEEE Robotics and Automation Letters*, 6(2): 2666–2673, 2021.

[10] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. Primal$_2$: Pathfinding via reinforcement and imitation multi-agent learning - lifelong. *IEEE Robotics and Automation Letters*, 6(2): 2666–2673, 2021. doi:10.1109/LRA.2021.3062803.

[11] A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov. Learn to follow: Lifelong multi-agent pathfinding with decentralized replanning. In *PRL Workshop Series – Bridging the Gap Between AI Planning and Reinforcement Learning*, 2023.

[12] K. Zhang, Z. Yang, and T. Başar. Decentralized multi-agent reinforcement learning with networked agents: Recent advances. *Frontiers of Information Technology & Electronic Engineering*, 22(6):802–814, June 2021. ISSN 2095-9230. doi:10.1631/FITEE.1900661. URL https://doi.org/10.1631/FITEE.1900661.

[13] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

[14] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.