MARBLER: An Open Platform for Standarized Evaluation of Multi-Robot Reinforcement Learning Algorithms

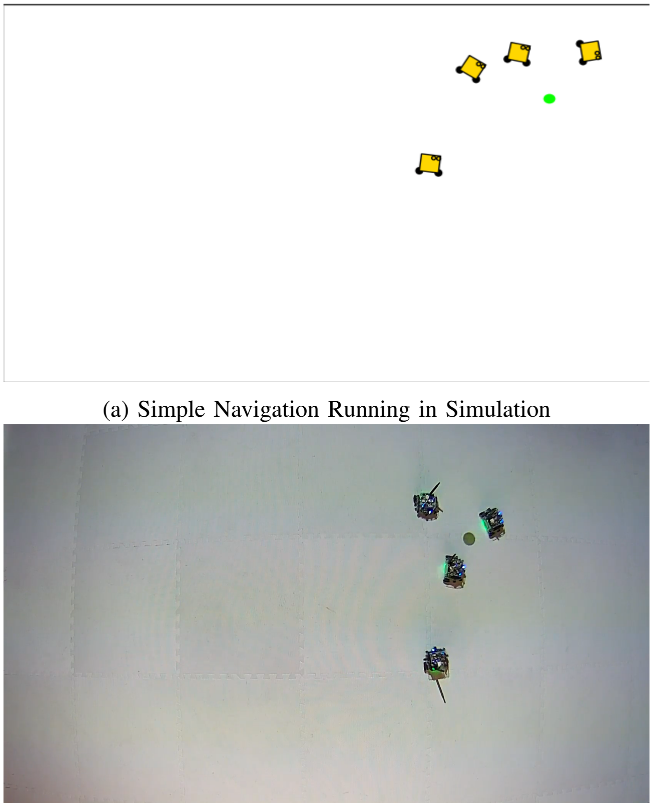
Supplementary Material

Reza J. Torbati, Shubham Lohiya[†], Shivika Singh[†], Meher S. Nigam, Harish Ravichandar

I. Detailed Scenario Information

In all scenarios, robots have 5 actions available to them: they can move left, right, up, down, or stop. After selecting an action, they move in the specified direction for a specified amount of time before selecting another action. MARBLER can support different action spaces but all current scenarios have this setup.

A. Simple Navigation



(a) Simple Navigation Running in Simulation

(b) Simple Navigation Running in Robotarium

1) Scenario Details: This is a fully homogeneous scenario where all robots try to get as close to a known location without colliding. The robots start at random locations in the left side of the environment and the destination will be a random location in the right side of the environment. This scenario is easy to learn, making it useful to use to quickly get robot up and running and to debug algorithms when they are not working. The robots' observations consist of their current location and the location of the destination point. Each episode ends after a specified number of steps.

Following the taxonomy described in [1], this scenario consists of robots that are physically identical and share the same objective which is to reach the target and avoid collision with the other robots. However, in our experiments, they have different behavior because we either trained the robots with agent ids or without parameter sharing.

2) Scenario Hyperparameters used for Hyperparameter Searches:

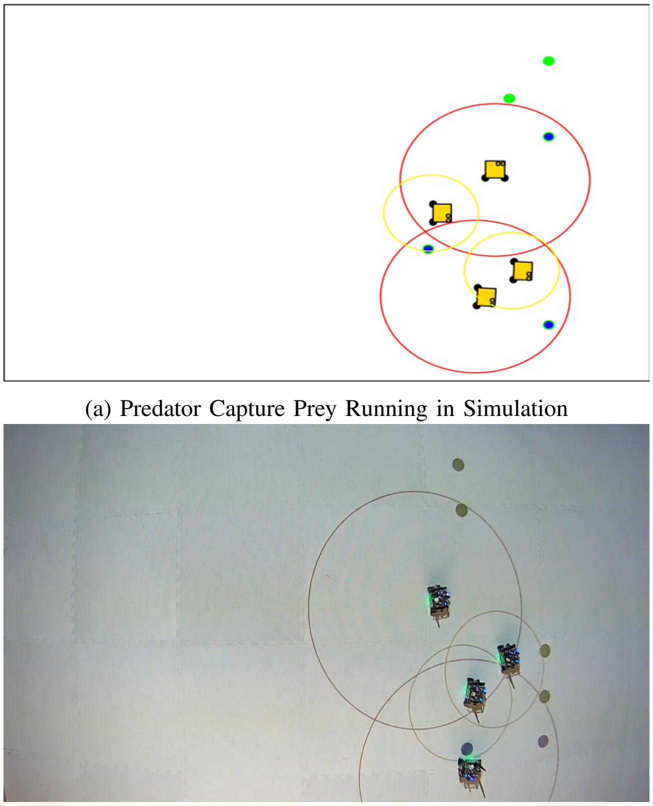
- *Number of Robots*: 4
- *Movement Speed*: $\sim$21cm/second
- *Episode Steps*: 51
- *Communication*: We gave the robots full communication so each robot had all other robots' observations concatenated together

B. Predator Capture Prey

1) Scenario Details: Inspired by the Predator Capture Prey scenario in [2], the objective of this scenario is to make heterogeneous robots collaborate to capture the prey as quickly as possible. There are n sensing robots, m capture robots and k prey. The robots start in random locations in the left side of the environment and the preys are randomly generated in the right side of the environment. The sensing robots have a sensing radius > 0 and the capture robots have a capture radius > 0. Both types of robots have their locations in their observation space. If there is a prey within a sensing robot's sensing radius, the prey will turn blue and the exact location of the prey will also be appended to the sensing robot's observation. However, the capture robot will never have the location of the prey appended to their observation so sensing robots and capture robots must work together to capture the prey. To capture a prey, a capture robot must stop once the prey is within its capture radius.

The episode ends once all preys have been captured or the episode times out. To incentivize the robots to capture the prey as quickly as possible, they take a small penalty every step the episode does not terminate. These robots have different capabilities and hence, differ in their behavior but share the same objective.

[†] Equal Contribution



(a) Predator Capture Prey Running in Simulation

(b) Predator Capture Prey Running in the Robotarium

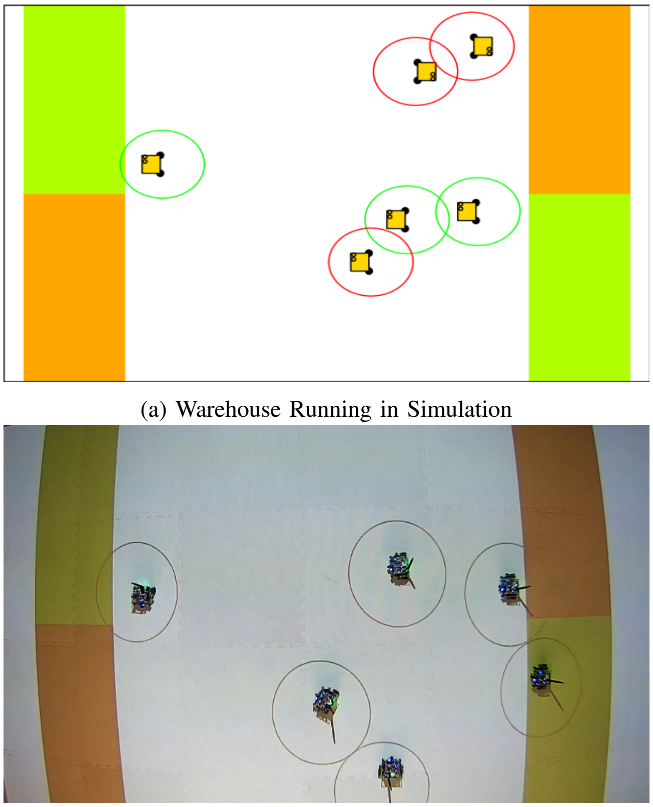
2) Selected Hyperparameters for Experiments:

- *Number of Robots*: 2 sensing, 2 capture
- *Number of Prey*: 6
- *Movement Speed*: ∼21cm/second
- *Sensing Radii*: 45cm
- *Capture Radii*: 25cm
- *Sensing Reward*: 1
- *Capture Reward*: 5
- *Step Penalty*: .05
- *Max Episode Steps*: 81
- *Communication*: We gave the robots full communication so each robot had all other robots' observations concatenated together.

C. Warehouse

1) Scenario Details: N/2 red robots and N/2 green robots start at random locations in the environment. The robots then navigate to the zone of their color on the right side of the environment to receive a load and obtain a small reward. They must then unload in their color zone on the left side of the environment which receives a large reward.

This scenario has fixed-length episodes so the robots only goal is to navigate to their respective zones as many times as possible. Because the optimal paths to the red and green loading zones intersect, optimal robots must learn to avoid collisions as the barrier certificates will slow them down significantly. Therefore, this scenario requires robots to learn multi-robot path finding [3].



(a) Warehouse Running in Simulation

(b) Warehouse Running in the Robotarium

The robots' observations consists of their current location and whether or not they are loaded. The robots are physically identical but with different goals and different behavior.

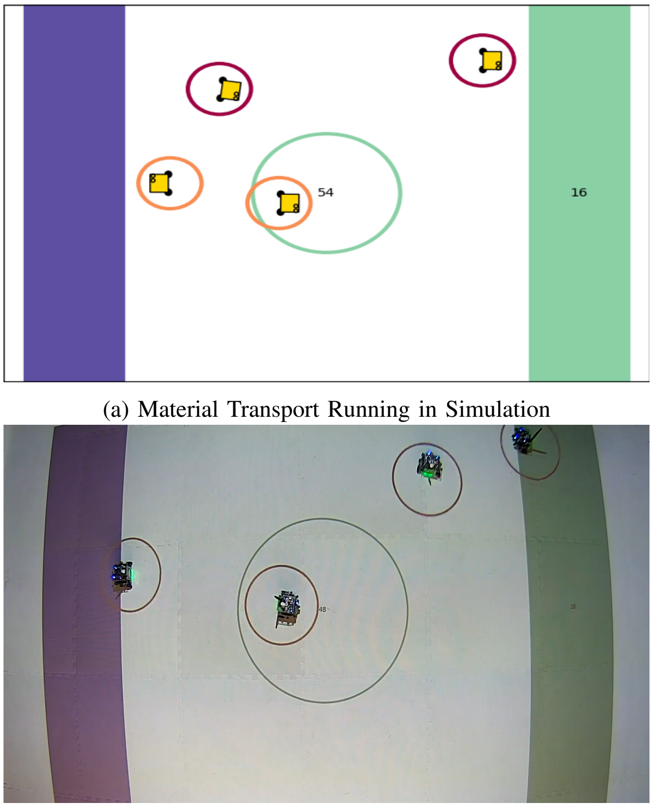
2) Selected Scenario Hyperparameters for Experiments:

- *Number of Robots*: 6
- *Movement Speed*: ∼21cm/second
- *Load Reward*: 1
- *Unload Reward*: 3
- *Episode Steps:* 101
- *Communication*: We gave the robots full communication so each robot had all other robots' observations concatenated together.

D. Material Transport

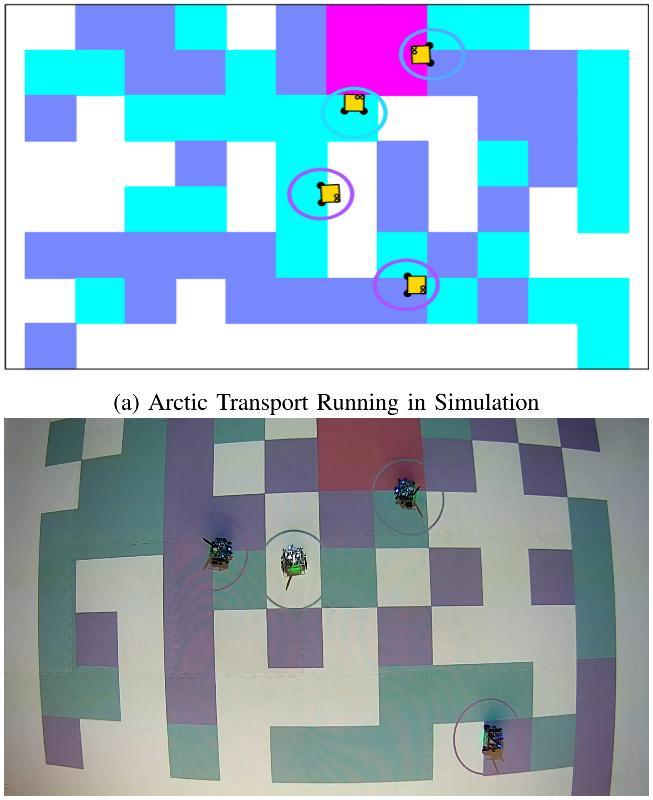
1) Scenario Details: N fast robots and M slow robots start at random locations within the purple zone and try to unload two zones as quickly as possible: one nearby and one far away. Both zones start with a random amount of material drawn from specified distributions. The robots must travel to the zone to receive a load and travel back to the purple area to unload. The fast robots can only carry a small load at once while the slow robots can carry a large load.

The episode ends once all of the material is transported to the purple zone or or the episode times out. To incentivize the robots to unload all of the material as quickly as possible, they take a small negative penalty every step the episode does not terminate. The robots observations consist of their current location and each zone's remaining load.



(a) Material Transport Running in Simulation

(b) Material Transport Running in the Robotarium



(a) Arctic Transport Running in Simulation

(b) Arctic Transport Running in the Robotarium

This scenario features two major challenges: first robots must navigate to the zones as quickly as possible meaning that they should not collide, again making this a multi robot path finding scenario. However, the path finding is easier in this scenario than in Warehouse because the robots can take routes that do not necessarily intersect. However, this scenario also requires robots to collaborate to make use of their respective capabilities to unload the zones as quickly as possible, meaning the scenario also requires the robots to effectively task allocate.

2) Scenario Hyperparameters for Experiments:

- *Number of Fast Robots*: 2
- *Number of Slow Robots*: 2
- *Fast Robot Movement Speed*: ∼18cm/second
- *Slow Robot Movement Speed*: ∼6cm/second
- *Fast Robot Loading Capacity*: 5
- *Slow Robot Loading Capacity*: 15
- *Load Reward*: .025 * Amount Loaded
- *Unload Reward*: .075 * Amount Unloaded
- *Step Penalty*: 0.1
- *Communication*: This scenario has full communication with limited bandwidth. Similar to how agents communicate in [4], robots can select a message from 0-3 to send to all other robots through their action space.

E. Arctic Transport

1) Scenario Details: Drones attempt to guide the water robots and ice robots to the goal location as quickly as possible over ground tiles (white), ice tiles (light blue), and water tiles (dark blue). Drones move fast over any tile, ice robots move fast over ice but slow over water, and water robots move fast over water but slow over ice. Ice and water robots move at the same speed on ground. Ice and water robots only know their location, the goal location, and the tile type that they are on. Drones know their location, the goal location, and the 8 surrounding tile types.

The robots start in the middle of the bottom row, which is always ground. The goal is randomly placed in the top row. Each remaining tile is randomly chosen to be one of the three tile types. Robots are only rewarded based on how far the ice/water robots are from the goal zone so optimal drones should guide the ice/water robots. This is a Multi-Robot Path Planning scenario [5] where drones must find optimal paths to the goal zone and communicate it to the ice/water robots.

The episode ends once both the water and the ice robots reaches the goal or the episode times out. To incentivize the robots to reach the goal as quickly as possible, they take a small penalty every step the episode does not terminate. These robots are physically different with different behavior but share the same objective.

2) Scenario Hyperparameters for Experiments:

- *Number of Drones*: 2
- *Number of Ice Robots*: 1
- *Number of Water Robots*: 1
- *Fast Movement Speed*: ∼25cm/second
- *Normal Movement Speed*: ∼17cm/second

- *Slow Movement Speed*: ~8.5cm/second
- *Reward*: -0.05*(Number of Ice or Water Robots that haven't reached the goal) + -0.075*($\sum$(Distance of Ice and Water Robots to Goal)^2)
- *Max Episode Steps:* 61
- *Communication*: We gave the robots full communication so each robot has all information from all other robots.

II. Additional Evaluation Returns During Training

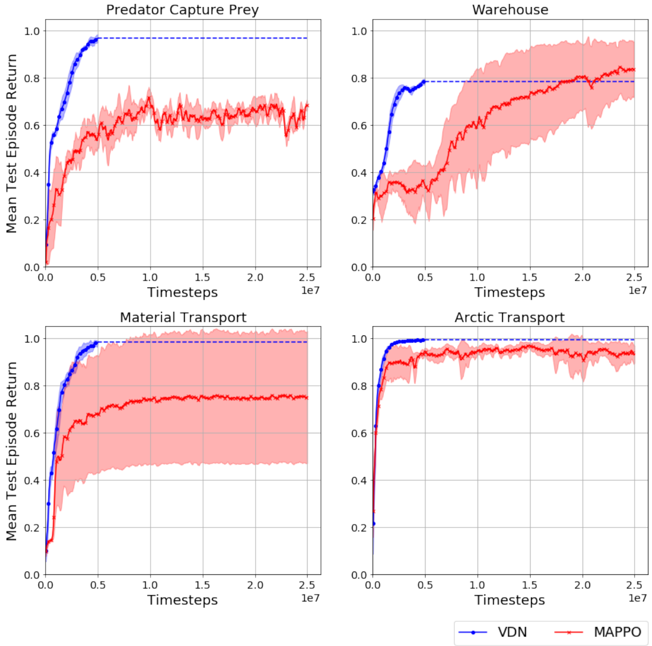


Fig. 6: The full 25 million timesteps for MAPPO. We also include VDN for comparison as VDN was the best performing algorithm for all scenarios after 5 million timesteps. MAPPO takes about the same amount of time as VDN to converge but is much more influenced by its seed than VDN and, except for warehouse, converges to a policy with lower performance.

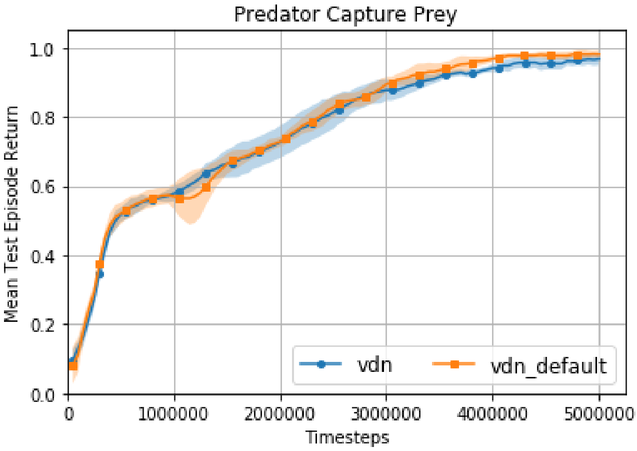


Fig. 7: Evaluation returns of VDN with the safe CBFs vs. VDN with the default CBFs during training.

III. Hyperparameter Details

A hyperparameter search was performed to find the best set of hyperparameters for each algorithm following the procedure used by EPyMARL [6]. We chose the Simple Navigation scenario to conduct the searches due to it's simplicity as it allows the algorithms to focus on learning the underlying dynamics, safety mechanisms, and communication which remains consistent across all scenarios.The search was performed across 2 seeds for each algorithm. For each set of hyperparameters we evaluated, we trained MAPPO for 4,000,000 time-steps and the value based algorithms for 400,000 time steps. We performed a grid-search over the range of hyperparameters as in Table I and used the evaluation return averaged across the seeds at each time-step to select the hyperparameters in Table II which are the final hyperparameters we selected for each algorithm. We selected the hyperparameters in Table I to search over based on the hyperparameters [6] analyzed.

	Algorithm			
Hyperparameters	**VDN**	**QMIX**	**QMIX_NS**	**MAPPO**
Learning Rate	{1,3,5} * 1e-4	{1,3,5} * 1e-4	{1,3,5} * 1e-4	{2,4} * 1e-4
Target Update	200/0.01	200/0.01	200/0.01	200/0.01
Hidden Dimension	64/128	64/128	64/128	64/128
Evaluation Epsilon	0.0/0.05	0.0/0.05	0.0/0.05	–
Epsilon Anneal	50k/200k	50k/200k	50k/200k	–
Entropy Coefficient	–	–	–	{1,10} * 1e-3
n-step	–	–	–	5/10
Standardize Rewards	Yes/No	Yes/No	Yes/No	Yes/No
Network Type	GRU	GRU	GRU	GRU

TABLE I: Hyperparameters searched over for each algorithm. For Taget Update, 200 is used for hard updates while 0.01 is used for soft updates.

	Algorithm			
Hyperparameters	**VDN**	**QMIX**	**QMIX_NS**	**MAPPO**
Learning Rate	0.0003	0.0001	0.0005	0.0002
Target Update	200	0.01	200	200
Hidden dimension	128	128	64	128
Evaluation epsilon	0.0	0.05	0.0	–
Epsilon Anneal	50k	50k	50k	–
Entropy Coefficient	–	–	–	0.01
n-step	–	–	–	10
Standardize Rewards	No	No	No	No
Network Type	GRU	GRU	GRU	GRU

TABLE II: Final hyperparameters selected for each algorithm.

IV. Limitations and Future Work

MARBLER's primary limitation is its training speed. On an Intel i7-12700H, we trained one seed of MPE's SimpleSpeakerListener scenario with QMIX for 1,000,000 time steps in 25 minutes, 57 seconds. PCP, in the same conditions with a single capture robot and a single predator robot, took 1 hour, 41 minutes, 19 seconds when training with CBFs enabled and 1 hour, 13 minutes, 45 seconds when training without CBFs enabled. This is a major limitation for MARBLER so future work should focus on making MARBLER faster.

We also did not benchmark MARBLER with GNN-based communication algorithms. To ensure the algorithms were fairly implemented, we used an unmodified version of EPyMARL which does not support GNNs. However, MARBLER can be used with GNNs and is currently being used in multiple studies that utilize GNN-based communication.

References

[1] M. Bettini, A. Shankar, and A. Prorok, "Heterogeneous multi-robot reinforcement learning," *arXiv preprint arXiv:2301.07137*, 2023.

[2] E. Seraj, Z. Wang, R. Paleja, D. Martin, M. Sklar, A. Patel, and M. Gombolay, "Learning efficient diverse communication for cooperative heterogeneous teaming," in *International conference on autonomous agents and multiagent systems*, 2022.

[3] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, 2019.

[4] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *CoRR*, vol. abs/1703.04908, 2017. [Online]. Available: http://arxiv.org/abs/1703.04908

[5] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Applied sciences*, vol. 9, no. 15, p. 3057, 2019.

[6] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. [Online]. Available: http://arxiv.org/abs/2006.07869