Scalable Multi-Agent Reinforcement Learning through Intelligent Information Aggregation

Siddharth Nayak [1] **Kenneth Choi** [1] **Wenqi Ding** [1] **Sydney Dolan** [1] **Karthik Gopalakrishnan** [2]
Hamsa Balakrishnan [1]

Abstract

We consider the problem of multi-agent navigation and collision avoidance when observations are limited to the local neighborhood of each agent. We propose InforMARL, a novel architecture for multi-agent reinforcement learning (MARL) which uses local information intelligently to compute paths for all the agents in a decentralized manner. Specifically, InforMARL aggregates information about the local neighborhood of agents for both the actor and the critic using a graph neural network and can be used in conjunction with any standard MARL algorithm. We show that (1) in training, InforMARL has better sample efficiency and performance than baseline approaches, despite using less information, and (2) in testing, it scales well to environments with arbitrary numbers of agents and obstacles. We illustrate these results using four task environments, including one with predetermined goals for each agent, and one in which the agents collectively try to cover all goals. Code available at https://github.com/nsidn98/InforMARL.

1. Introduction

Reinforcement Learning (RL) has seen wide-ranging successes recently in high-dimensional robot control (Lillicrap et al., 2016; Levine et al., 2015), solving physics-based control problems (Heess et al., 2015), playing Go (Silver et al., 2016), Chess (Silver et al., 2017) and Atari video games (Mnih et al., 2013; 2015), etc. However, challenges remain in many real-world applications in which the tasks cannot be handled by a single agent, e.g., multi-player games, search-and-rescue drone missions, etc. (Gronauer & Diepold, 2022). In such cases, multiple agents may need to work together and share information in order to accomplish the task (Tan, 1993b). Naïve extensions of single-agent RL algorithms to multi-agent settings do not work well because of the non-stationarity in the environment, i.e., the actions of one agent affect the actions of others (Tan, 1993a; Tampuu et al., 2015). Furthermore, tasks may require cooperation among the agents. Classical approaches to optimal planning may (1) be computationally intractable, especially for real-time applications, and (2) be unable to account for complex interactions and shared objectives between multiple agents. The ability of RL to learn by trial-and-error makes it well-suited for problems in which optimization-based methods are not effective. In particular, multi-agent reinforcement learning (MARL) approaches may be suitable in these situations due to their fast run-times and superior performance, and their ability to model shared goals between agents using appropriate reward structures.

In this paper, we focus on multi-agent navigation and collision avoidance problems in which there are N agents trying to cooperate with each other to collectively solve a task in a 2D environment with static and/or dynamic obstacles. The rewards are shared across all agents in these collaborative environments. We assume that an agent can only sense the presence of obstacles or other agents within a certain limited radius r. The overarching objective is for all the agents to complete their tasks in the shortest time possible, while avoiding collisions with other agents and obstacles. This problem setting is quite general and arises in many contexts, *e.g.*, search-and-rescue robot teams (Kumar et al., 2004), environmental monitoring (Dunbabin & Marques, 2012), and drone delivery systems (Zhang et al., 2014; Dorling et al., 2017; Hii et al., 2019).

MARL-based techniques have achieved significant successes in recent times, *e.g.*, DeepMind's AlphaStar surpassing professional level players in StarCraft II (Vinyals et al., 2019), OpenAI Five defeating the world-champion in Dota II (Berner et al., 2019), etc. The performance of many of these MARL algorithms depends on the amount of information included in the state given as input to the neural networks (Yu et al., 2022). In many practical multi-agent

[1]Massachusetts Institute of Technology, Cambridge, USA [2]Stanford University, Stanford, USA. Correspondence to: Siddharth Nayak <sidnayak@mit.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

scenarios, each agent aims to share as little information as possible to accomplish the task at hand. This structure naturally arises in many multi-agent navigation settings, where agents may have a desired end goal but do not want to share their information due to communication constraints or proprietary concerns (Rendleman & Mountin, 2015; Weeden et al., 2019). These scenarios result in a decentralized structure, as agents only have locally available information about the overall system's state. In this paper, we focus on the question: "*Can we train scalable multi-agent reinforcement learning policies that use limited local information about the environment to perform collision-free navigation effectively?*"

We propose *InforMARL*, an approach for solving the multi-agent navigation problem using graph-reinforcement learning. The main features of our approach are that it: (1) uses a graph representation of the navigation environment which enables local information-sharing across the edges of the graph; (2) transfers well to different numbers of agents; and (3) achieves better sample complexity in training compared to prior approaches by aggregating relevant local information from neighbors in the underlying graph. More broadly, our work (1) demonstrates that graphs provide a valuable abstraction for multi-agent navigation environments; (2) highlights that more information (i.e., global information as states) may not necessarily improve performance, and can, in fact, overwhelm the RL agent networks and lead to increased sample complexity; and (3) shows how graph architectures can identify the most valuable information for navigation from local observations to improve performance and scalability.

2. Related Work

Multi-agent navigation problems arise in a number of other contexts, *e.g.* multi-robot navigation. Optimization-based (Boldrer et al., 2020) and trajectory-based (Phillips & Likhachev, 2011; Luders et al., 2011; Ángel Madridano et al., 2021) approaches have been used for multi-robot navigation. However, the former often has long computation times making real-time execution infeasible, while the latter can encounter the "freezing robot problem" in dense environments due to a large portion of the statespace being marked as unsafe. We therefore focus on MARL approaches.

2.1. Scaling MARL

Research on scaling MARL algorithms has broadly followed two main themes: (1) decentralized execution, and (2) transferring learning between scenarios.

Decentralized MARL: Centralized-training-decentralized-execution (CTDE) is a popular approach to improve scaling. CTDE frameworks typically use actor-critic methods (Konda & Tsitsiklis, 1999), where the training step uses a centralized critic that incorporates global information from all actors. During execution, the agents use their own actor networks to select their actions in a decentralized manner. MADDPG (Lowe et al., 2017a) builds upon DDPG (Lillicrap et al., 2016) by learning a centralized critic that is provided the joint state and actions of all agents. MATD3 (Ackermann et al., 2019) uses a double-centralized critic model, reducing the over-estimation bias. (Yu et al., 2022) show the effectiveness of PPO in several standard multi-agent environments. VDN (Sunehag et al., 2017) decomposes a centralized value function to a sum of individual agent-specific functions. Q-Mix (Rashid et al., 2018) improves upon this by imposing a monotonicity requirement on agents' individual value functions and using a learnable mixing of the individual functions. These MARL algorithms perform well in the navigation environment when they know the positions of all entities in the environment. But, as we will demonstrate, they fail to learn when that information is restricted to just local neighborhoods around the agents.

Transferability in MARL: It is desirable to have MARL formulations where the number of agents/entities in the environment doesn't hinder the performance of the model. Most of the previous works using MARL for the navigation task require concatenation of observations of other entities in the environment to be able to learn meaningful policies. As the neural network size depends on the state input dimensions used while training, the learned policy fails to work in scenarios with a different number of entities in the environment. With the recent success of graph neural networks, many recent works have focused on leveraging the inherent graph structure present in multi-agent problems and tackling the issue of transferability. Zhou et al. (Zhou et al., 2021) create a neighborhood graph according to how close the entities are to each other and use imitation learning to imitate a greedy behavior for the target-coverage problem (Tokekar et al., 2014; Dames et al., 2017). Similarly, Khan et al. (Khan et al., 2019) use graph neural networks with vanilla policy gradient (Sutton et al., 1999) for the formation flying task (Turpin et al., 2012a;b). They show that dynamic graphs have worse performance than static graphs due to the large number of possible graphs the model has to learn. At the beginning of each episode, their model determines the connectivity of the agents with each other and uses the same graph over the whole episode. This works well for the formation flying task as the graph structure does not change much if the agents fly in the same formation over an episode. DGN (Jiang et al., 2020) combines a graph convolutional neural network (Kipf & Welling, 2016) architecture with multi-head attention (Vaswani et al., 2017) for a variety of multi-agent environments, including 2D coverage and tracking. The DGN architecture assumes that each agent communicates with its three closest neighbors. Communi-

cating with the three closest agents leads to a graph that is always connected. However, the three-agent connectivity assumption is highly restrictive in real life applications, as communication is generally restricted by mutual separation due to hardware constraints. Similarly, (Liu et al., 2019; Chen et al., 2021) use an equivalent DGN-like architecture for multi-agent control.

We use a distance-based agent-entity graph similar to Entity-Message Passing (EMP) (Agarwal et al., 2019). However, the EMP architecture assumes that the agents have access to the positions of all entities in the environment at the beginning of the episode, which is not possible if there are occluded static or dynamic obstacles. By contrast, our model does not make this assumption.

2.2. Information Sharing for MARL

For various cooperative tasks where explicit coordination is required to solve the task, enabling communication to share information across the agents helps with the performance. In CommNet (Sukhbaatar et al., 2016), the authors introduce a model to learn a differentiable communication protocol between multiple agents. However, it does not explicitly model the interactions between the agents but rather averages the states of all the neighboring agents. VAIN (Hoshen, 2017) improves upon CommNet by using an exponential kernel-based attention to choose specific messages from other agents to attend to. Similarly, ATOC (Jiang & Lu, 2018) and TarMAC (Das et al., 2018) use an attention mechanism for communication among agents but without any restrictions on which agent can communicate with which others, leading to centralization during execution. GAXNET (Yun et al., 2021) also uses an attention mechanism (Vaswani et al., 2017), but additionally allows for the exchanging of weights with other agents to reduce the attention mismatch between them. While their model is shown to work well for navigation, it requires the maximum number of agents in the environment to be fixed before training. We also use an attention mechanism for inter-agent communication. However, we do not have any message passing between objects in the environment (also called 'entities') and agents; instead, the agents themselves do all the computation with local information of the entities' states.

3. Description of InforMARL

Our MARL framework for navigation, InforMARL, consists of four modules, as shown in Figure 1 and described below.

3.1. Environment

Every object in the environment (also known as an 'entity') is assumed to be either an agent, an obstacle, or a goal/landmark. We define an agent-entity graph with respect to agent i at each time-step t, as $g_t^{(i)} \in \mathcal{G} : (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ is an entity in the environment. The variable $\texttt{entity\_type(j)} \in \{\texttt{agent}, \texttt{obstacle}, \texttt{goal}\}$ determines the type of entity at node j. There exists an edge $e \in \mathcal{E}$ between an agent and an entity if they are within a 'sensing radius' ρ of each other. The agent-agent edges are bidirectional, whereas the agent-non-agent edges are unidirectional (i.e., messages can only be passed from the non-agent entity to the agent). In other words, a unidirectional edge is equivalent to the agent sensing a nearby entity's state, while a bidirectional edge is equivalent to a communication channel between agents. This structure is similar to the agent-entity graph defined in (Agarwal et al., 2019), but without the assumption that all agents have access to the positions of all entities at the beginning of each episode. Note that our formulation also supports cases where disconnected sub-graphs are formed due to the positioning of the entities in the environment.

The corresponding Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Puterman, 1994; Kaelbling et al., 1998; Oliehoek & Amato, 2016) is characterized by the tuple $\langle N, S, O, \mathcal{A}, \mathcal{G}, P, R, \gamma \rangle$, where N is the number of agents, $s \in S = \mathbb{R}^{N \times D}$ is the state space of the environment and D is the dimension of the state, and $o^{(i)} = O(s^{(i)}) \in \mathbb{R}^d$ is the local observation for agent i, where $d \leq D$ is the observation dimension. $a^{(i)} \in \mathcal{A}$ is the action space for agent i and the joint action for all N agents is given by $A = (a^{(1)}, \cdots, a^{(N)})$. Specifically, $a^{(i)}$ is a one-hot vector of size equal to the number of possible actions. $g^{(i)} \in \mathcal{G}(s; i)$ is the graph network formed by the entities in the environment with respect to agent i. $P(s'|s, A)$ is the transition probability from s to s' given the joint action A. $R(s, A)$ is the joint reward function. $\gamma \in [0, 1)$ is the discount factor. The MARL training process seeks to find an optimal policy, $\Pi = \left(\pi^{(1)}, \cdots, \pi^{(N)}\right)$, where each agent uses a policy $\pi_\theta^{(i)}\left(a^{(i)}|o^{(i)}, g^{(i)}\right)$ parameterized by θ to determine its action $a^{(i)}$ from its local observation $o^{(i)}$ and the graph network $g^{(i)}$ that it is a part of, while optimizing the discounted accumulated reward $J(\theta) = \mathbb{E}_{A_t, s_t}\left[\sum_t \gamma^t R\left(s_t, A_t\right)\right]$.

Agent i's local observation $o^{(i)}$ consists of its position and velocity in a global frame of reference and the relative position of the agent's goal with respect to its position. Each node j on the graph $g^{(i)}$ has node features $x_j = [p_i^j, v_i^j, p_i^{\text{goal},j}, \texttt{entity\_type(j)}]$ where $p_i^j, v_i^j, p_i^{\text{goal},j}$ are the *relative* position, velocity, and position of the goal of the entity at node j with respect to agent i, respectively. If node j corresponds to a (static/dynamic) obstacle or a goal, we set $p_i^{\text{goal},j} \equiv p_i^j$. Each edge e_{ij} has an associated edge feature given by the Euclidean distance between the entities i and j. For processing the $\texttt{entity\_type}$

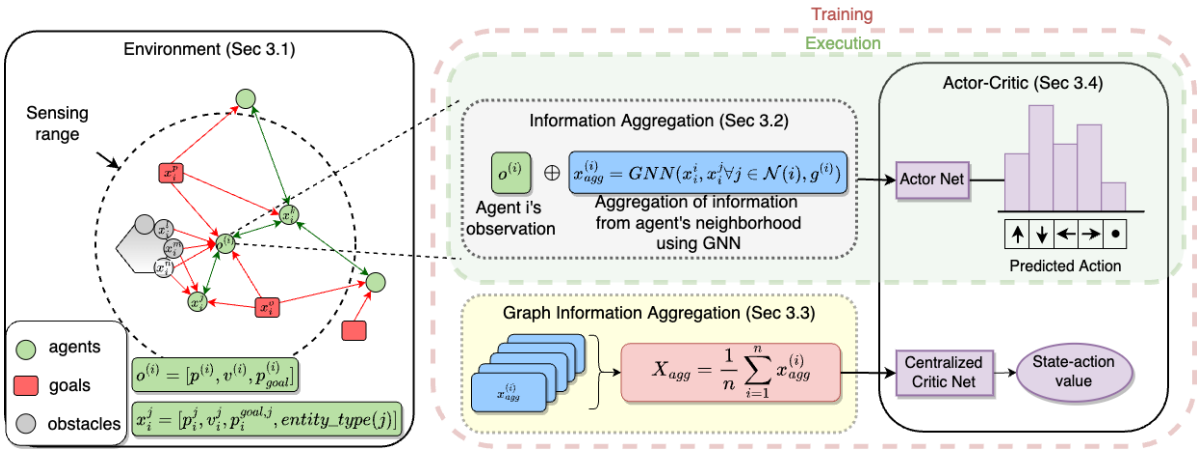


Figure 1. Overview of InforMARL. (i) Environment: The agents are depicted by green circles, the goals are depicted by red rectangles, and the unknown obstacles are depicted by gray circles. $x^{(i)}_{agg}$ represents the aggregated information from the neighborhood, which is the output of the GNN. A graph is created by connecting entities within the sensing-radius of the agents. (ii) Information Aggregation: Each agent's observation is concatenated with $x^{(i)}_{\text{agg}}$. The inter-agent edges are bidirectional, while the edges between agents and non-agent entities are unidirectional. (iii) Graph Information Aggregation: The aggregated vector from all the agents is averaged to get X_{agg}. (iv) Actor-Critic: The concatenated vector $[o^{(i)}, x^{(i)}_{\text{agg}}]$ is fed into the actor network to get the action, and X_{agg} is fed into the critic network to get the state-action values.

categorical variable, we experimented with both using an embedding layer (Mikolov et al., 2013) and using one-hot encoding. No significant performance advantage with one method over the other was found, so we chose to use the embedding layer. Further analysis revealed that the learned embedding vectors for `entity_type` were equidistant from each other when visualized in 2D. This was to be expected because each of the entity types $(agent, obstacle, goal)$ are equally distinct from another. Future work could include further refinement of `entity_type`, *e.g.*, adversarial vs. cooperative obstacles, static vs. dynamic obstacles, etc.

We adopt a similar reward function as used in multi-agent particle environment (MAPE) (Lowe et al., 2017b) where the joint reward function is defined as $R(s_t, A_t) = \sum_{i=1}^{N} r^{(i)}_t$, which encourages cooperation among all agents. Here $r^{(i)}_t$ is each agent's reward at timestep t and depends on the scenario. Details about the reward functions for the different task environments can be found in Appendix B.

3.2. Information Aggregation

To infer information about the local neighborhood around each agent i, we use a graph neural network (GNN) with a message passing framework (Gilmer et al., 2017). Specifically, we use Unified Message Passing Model (UniMP) (Shi et al., 2020), a variant of a graph transformer (Vaswani et al., 2017; Dwivedi & Bresson, 2020) where each layer update is defined as $x'_i = W_1 \cdot x_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_2 \cdot x_j$, where x_k are the node features in the graph, $\mathcal{N}(i)$ is the set of nodes which are connected to node i, W_k are learnable weight matrices and the attention coefficients $\alpha_{i,j}$ are computed via multi-head dot product attention:

$$\alpha_{i,j} = \text{softmax}\left(\frac{(W_3 \cdot x_i)^T (W_4 \cdot x_j + W_5 \cdot e_{ij})}{\sqrt{c}}\right) \quad (1)$$

where e_{ij} are edge features for the edge connecting nodes i and j, and c is the output dimension for that specific layer.

The attention mechanism allows the agents to selectively prioritize messages coming from their neighbors according to their importance. We use multiple layers of this message-passing so that information can be propagated between agents that are higher-order neighbors with each other. For each agent i, this module aggregates information from the neighboring nodes in the graph into a fixed-sized vector $x^{(i)}_{\text{agg}}$. The concatenated vector $[o^{(i)}, x^{(i)}_{\text{agg}}]$ is given as the input to the actor network. This architecture allows InforMARL to dynamically adapt to a changing number of entities in the environment while remaining invariant to the permutation of the observed entities.

3.3. Graph Information Aggregation

While training a model in the CTDE setting, the critic generally gets the state-action pairs of all individual agents in the environment as a concatenated vector. To make the training transferable to a variable number of agents and to aid with curriculum learning (Narvekar et al., 2020; Taylor & Stone, 2009; Lazaric et al., 2008), we replace this concatenation with a graph information aggregation module. This module

is similar to the 'Information Aggregation' one in which a GNN aggregates information from the agent's neighbors. A global mean pooling operator, $X_{\text{agg}} = \frac{1}{N}\sum_{i=1}^{N} x_{\text{agg}}^{(i)}$, is applied to aggregate the updated node features in the graph. Note that X_{agg} is a vector of fixed size independent of the number of agents, which is not the case when concatenating the state action-pairs of all individual agents. This vector is then given as input to the critic network.

3.4. Actor-Critic Networks

The actor and critic networks can be either a multi-layer perceptron (MLP) or a recurrent neural network (RNN) (Hausknecht & Stone, 2015), using either LSTMs (Hochreiter & Schmidhuber, 1997) or GRUs (Cho et al., 2014). Our proposed information aggregation method can be used in conjunction with any standard MARL algorithm (*e.g.*, MADDPG (Lowe et al., 2017a), MATD3 (Ackermann et al., 2019), MAPPO (Yu et al., 2022), QMIX (Rashid et al., 2018), VDN (Sunehag et al., 2017), etc.).

4. Experiments

Environment descriptions: We evaluate our proposed model on four different navigation tasks by modifying the MAPE (Lowe et al., 2017b). In all these environments, N agents move around in a 2D space following a double integrator dynamics model (Rao & Bernstein, 2001). Each agent has a discrete action space where it can control unit acceleration and deceleration in the x- and y- directions.

1. **Target**: Each agent tries to reach its preassigned goal while avoiding collisions with other entities in the environment.
2. **Coverage** (Tokekar et al., 2014; Dames et al., 2017): Each agent tries to go to a goal while avoiding collisions with other entities, and ensuring that no more than one agent reaches the same goal.
3. **Formation** (Agarwal et al., 2019): There is a single landmark (the counterpart of a goal for this task), and the agents try to position themselves in an N-sided regular polygon with the landmark at its centre.
4. **Line** (Agarwal et al., 2019): There are two landmarks, and the agents try to position themselves equally spread out in a line between the two.

We first focus on the *Target* task environment in this section, and present the performance of InforMARL on the other tasks in Section 4.4. More details about the environments can be found in Appendix B.

Implementation specifications: We chose to use MAPPO (Yu et al., 2022) as the base MARL algorithm for InforMARL because it was found to be the best performing of the standard MARL baselines in the 3 agent-3 obstacle *Target* environment. We implemented InforMARL by modifying the official codebase for MAPPO in PyTorch. The codebase links to our baseline implementations can be found in Appendix A. We used the official implementations for most of the baselines considered in Section 4.2. We prefix the algorithm name with 'R' to denote the recurrent neural network (RNN) version of the algorithm (*e.g.* MAPPO-RMAPPO, etc.). We use the same hyperparameters as used in the experiments for MAPPO, and do not perform parameter tuning on InforMARL for any of the MAPPO-based parameters. The hyperparameters used in our experiments can be found in Appendix C.

Amount of information available to agents: The amount of information available to each agent determines whether or not it can learn a meaningful policy. Although having more information generally translates to better performance, it does not necessarily scale well with the number of agents. Prior works (Yu et al., 2022; Lowe et al., 2017a) have typically used a naïve concatenation of the states of all agents or entities in the environment fed into a neural network. Such an approach scales poorly (the network input size is determined by the number of agents) and does not transfer well to scenarios with a different number of agents than the training environment. We vary the amount of information available to agents by defining three information modes:

- *Local:* In the local information mode, $o_{\text{loc}}^{(i)} = [p^{(i)}, v^{(i)}, p_{\text{goal}}^{(i)}]$ where $p^{(i)}$ and $v^{(i)}$ are the position and velocity of agent i in a global frame, and $p_{\text{goal}}^{(i)}$ is the position of the goal relative to the agent's position. InforMARL uses this information mode.
- *Global:* Here, $o_{\text{glob}}^{(i)} = [p^{(i)}, v^{(i)}, p_{\text{goal}}^{(i)}, p_{\text{other}}^{(i)}]$, where $p_{\text{other}}^{(i)}$ comprises of the relative positions of all the other entities in the environment. The scenarios defined in the MAPE (and consequently, other approaches that use MAPE) use this type of information mode unless explicitly stated otherwise.
- *Neighborhood:* Here, agent i observes $o_{\text{nbd}}^{(i)} = [p^{(i)}, v^{(i)}, p_{\text{goal}}^{(i)}, p_{\text{other}}^{(i)}]$, where $p_{\text{other}}^{(i)}$ comprises of the relative positions of all other entities which are within a distance `nbd-dist` of the agent. The maximum number of entities within the neighborhood is denoted `max-nbd-entities`, and so the dimension of the observation vector is fixed. If there are fewer than `max-nbd-entities` within a distance `nbd-dist` of the agent, we pad this vector with zeros.

4.1. A Motivating Experiment

The *local* or *neighborhood* information modes are transferable to scenarios with a different number of entities in the environment. By contrast, the *global* information mode

is not transferable to other scenarios. Figure 2 shows the rewards obtained during training using the three information modes in the *Target* task environment.

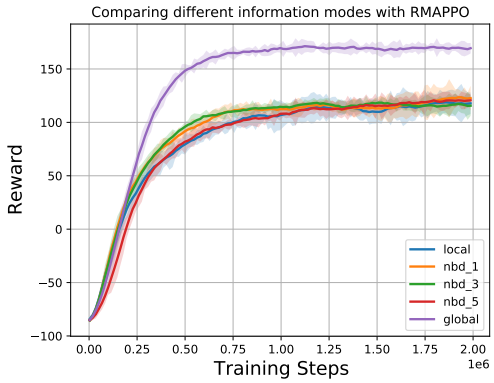


Figure 2. RMAPPO with the local, neighborhood (with 1, 3, and 5 `max-nbd-entities`), and global information given as states. The plots show the rewards during training for the 3 agent-3 obstacle *Target* task environment. Comparing the global information mode to the others, we see that merely providing local information and a naïve concatenation of neighborhood information is not sufficient to learn an optimal policy.

We see in Figure 2 that the policy learned with global information is better than the policies learned with local or neighborhood information. This is because it has the information necessary to take optimal actions. In the 'nbd_5' scenario, if all entities are within the `nbd_dist` ball, then $o^{(i)}_{\text{nbd}} \equiv o^{(i)}_{\text{glob}}$, since there are only five entities in the environment apart from the agent itself. Although the 'nbd_5' scenario is almost similar to the global information mode, the performances are not similar: the global information mode achieves much higher rewards. This behavior is because the observation $o_{i,\text{nbd}}$ can change temporally from having information about an entity when it lies within a `nbd_dist` ball of the agent, and then getting padded with zeros when it is out of the ball. This inconsistency in the amount of information accessible to the agent at every time step causes a significant performance difference between the policies learned with global and neighborhood information modes. This experiment motivates us to find a way in which more information can be leveraged (similar to the global case), but in a manner that does not suffer from the performance shortcomings of the neighborhood or local information modes.

4.2. Comparison of InforMARL with Other Baselines

As shown in Section 4.1, using local information modes or naïvely concatenating neighborhood entity information is not sufficient to learn optimal policies. In this section, we demonstrate that InforMARL can effectively learn policies for navigation given local information. We then compare its performance in the *Target* environment with prior deep MARL approaches. Specifically, we consider the following methods as baselines for comparison.

1. **Graph Policy Gradient (GPG) (Khan et al., 2019):** GPG uses a graph convolutional neural network (GCN) (Kipf & Welling, 2016) to parameterize policies for agents. The authors use the policy gradient method (Sutton et al., 1999) as the base MARL algorithm. We perform experiments with both dynamic and static graphs. *(Note: It was shown in (Khan et al., 2019) that using a static graph constructed at the beginning of the episode was better than using a dynamic graph.)*
2. **Graph Convolutional Reinforcement Learning (DGN, DGN+ATOC) (Jiang & Lu, 2018; Jiang et al., 2020):** Similar to GPG, these methods use GCNs to capture interactions between the agents in the environment. A key difference between InforMARL and these two methods (GPG and DGN) is that while the latter two approaches consider only agents in their message-passing graph, InforMARL also includes other (non-agent) entities in the graph.
3. **Entity Message Passing (EMP) (Agarwal et al., 2019):** Similar to InforMARL, EMP uses an agent-entity graph. However, in contrast to InforMARL, EMP assumes that agents know the positions of all entities in the graphs (i.e., global information) at the beginning of the episode.
4. **Other standard MARL Algorithms:** Finally, we compare InforMARL with standard MARL algorithms, namely, MADDPG (Lowe et al., 2017a), MATD3 (Ackermann et al., 2019), QMIX (Rashid et al., 2018), VDN (Sunehag et al., 2017) and MAPPO (Yu et al., 2022). In each case, we also consider the recurrent neural network versions. We focus on results for the global information modes, we found that these methods did not learn well with just local information.

Figure 3 shows the training performance of InforMARL and the best-performing of the baselines mentioned above for the *target* task environment. For ease of visualization, we plot only the four best-performing methods with global and local information, respectively. Each line corresponds to the mean and standard deviation over five random seeds. We consider scenarios with $N = \{3, 7, 10\}$ agents. Comparisons to other baselines can be found in Appendix D, and ablation studies for varying sensing radii in Appendix E.

Figure 3 illustrates that only using RMAPPO (i.e., the RNN version of MAPPO) or InforMARL, agents are able to learn to navigate and get to their goals. However, unlike RMAPPO which requires global information, InforMARL achieves this with just local information. Furthermore, InforMARL requires a similar number of training steps as RMAPPO, despite having access to much less information.

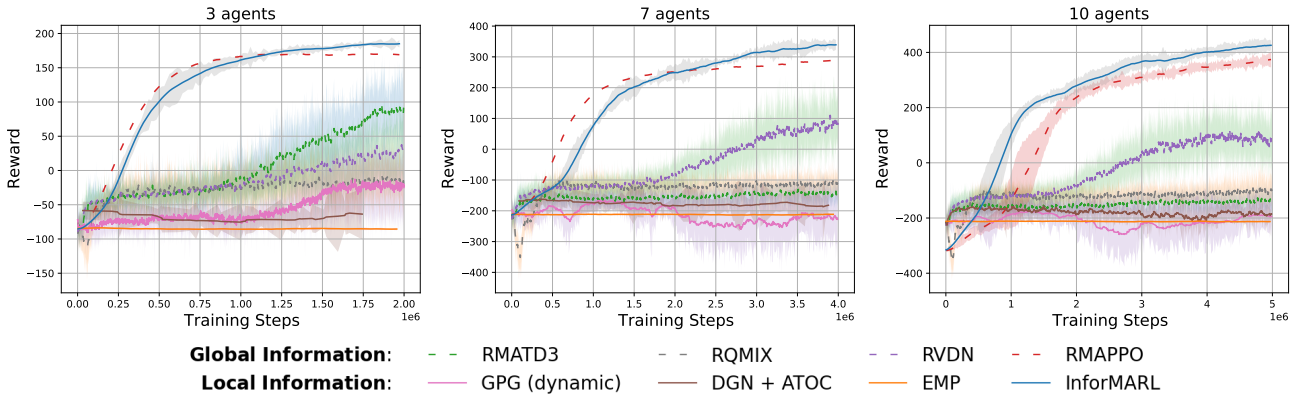


Figure 3. Comparison of the training performance of InforMARL with the best-performing baselines using global and local information in the *Target* task environment. The means and standard deviations of the rewards over training with five random seeds are shown. InforMARL significantly outperforms most baseline algorithms. Although RMAPPO has similar performance, it requires global information. Appendix D presents a complete comparison to more baselines.

Algorithm	**Info mode**	$N = 3$				$N = 7$				$N = 10$			
		Reward	T	# col	$S\%$	Reward	T	# col	$S\%$	Reward	T	# col	$S\%$
RMADDPG	Global	10.73	0.75	1.72	23	−122.07	0.95	4.61	3	-127.98	1.00	7.39	0
RMATD3	Global	105.49	0.51	1.62	67	−128.93	0.96	3.67	6	-131.72	0.99	5.94	1
RQMIX	Global	19.21	0.77	0.83	28	−83.41	0.85	5.83	12	-76.98	0.96	8.65	2
RVDN	Global	64.04	0.62	0.57	45	140.94	0.62	3.42	47	157.63	0.64	4.93	43
RMAPPO	**Global**	**173.13**	**0.41**	**0.67**	**96**	**327.39**	**0.44**	**4.29**	**88**	**366.81**	**0.44**	**6.93**	**79**
GPG (dyn.)	Local	−46.27	0.87	0.21	8	−165.91	1.00	1.57	3	-173.53	1.00	2.24	0
DGN+ATOC	Local	67.70	0.66	0.72	35	−189.61	0.97	1.03	0	-201.01	1.00	1.97	0
EMP	Local	−83.96	0.98	0.72	6	−211.90	0.98	2.63	0	-209.90	1.00	94.12	0
InforMARL	**Local**	**205.24**	**0.17**	**1.45**	**100**	**399.01**	**0.37**	**3.72**	**100**	**429.14**	**0.39**	**4.73**	**100**

Table 1. Comparison of InforMARL with the best-performing baseline methods, for the *Target* task environment with 3, 7, and 10 agents, averaged over 100 test episodes. See Appendix D for comparisons to other baseline methods.

We present the following metrics in Table 1. The results represent an average over 100 test episodes.

1. The total rewards obtained by the agents during an episode A higher value corresponds to better performance.
2. The fraction of an episode that the agents take on average to get to the goal, denoted T. If an agent does not reach its goal, then T is set to be 1 (lower is better).
3. Percent of episodes in which all agents are able to get to their goals, denoted $S\%$ (higher is better).
4. The total number of collisions (both agent-agent + agent-obstacle) that agents had in an episode, denoted # col. The lower this metric, the better the performance of the algorithm.

Although having a smaller number of collisions is better, the policies of some of the baseline algorithms do not significantly move the agents from their initial position after training and hence do not get to the goal. This leads to them having a lower number of collisions. Hence, this metric should be judged with the success rate in context.

The graph-based methods, namely GPG (static and dynamic), DGN (+ATOC), and EMP do not learn effectively with local information modes. Although both GPG and DGN use GCNs, they do not perform as well as InforMARL because they use only agent-agent and not agent-entity graphs. The lack of information about non-agent entities means that agents cannot maneuver through the environment to avoid collisions.

In contrast to the results showed in (Khan et al., 2019), our results show that the dynamic graph version of GPG is slightly better than the static graph one. A possible reason for this discrepancy is that the fixed formation environment considered in (Khan et al., 2019) is more amenable to the use of static graphs than the navigation environment. EMP, which uses a similar agent-entity graph as ours, fails to learn because of the strong assumption of having access to the positions of all entities in the environment at the beginning of the episode. The original implementation of EMP (and the associated environments) included information about all the entities other than agents in the observation vector.

4.3. Scalability of InforMARL

To evaluate the scalability of InforMARL with minimum information, we perform experiments in the *Target* environment by testing the models in scenarios with a different number of agents from those they were trained on.

Test \ Train		$n = 3$	$n = 7$	$n = 10$
$m = 3$	Reward/m	63.21	63.25	62.87
	T	0.39	0.40	0.40
	(# col)/m	0.40	0.46	0.49
	$S\%$	100	100	99
$m = 7$	Reward/m	61.16	62.23	61.32
	T	0.38	0.40	0.40
	(# col)/m	0.74	0.66	0.70
	$S\%$	100	100	100
$m = 10$	Reward/m	58.59	58.23	58.67
	T	0.38	0.40	0.39
	(# col)/m	0.95	0.88	0.87
	$S\%$	100	99	100
$m = 15$	Reward/m	53.19	53.46	54.21
	T	0.39	0.40	0.40
	(# col)/m	1.28	1.21	1.20
	$S\%$	100	99	99

Table 2. Test performance of InforMARL for the *Target* task, when trained on scenarios with n agents and tested with m agents in the environment.

Table 2 shows the results of testing InforMARL trained on n agents and tested on m agents. Each scenario is tested over 100 episodes. The number of obstacles in the environment is randomly chosen from $(0, 10)$ at the beginning of the episode. Based on the findings presented in Table 1, we did not test the scalability of other methods since they did not perform well in the local information mode, or could not handle varying numbers of entities in the environment.

We see from Table 2 that InforMARL is able to achieve a success rate of 100% for almost all the scenarios when the number of agents in the environment as varied. Furthermore, with InforMARL, the agents are able to get to their goals within $T \sim 0.39$ of the episode length in all scenarios. The number of collisions per agent increases, which is to be expected as the environment becomes denser. We believe that this can be remedied by using a stricter penalty for collisions. Alternatively, control barrier functions for satisfying safety constraints could be used to provide a formal safety guarantee in the MARL setting (Qin et al., 2021).

4.4. Performance in Other Task Environments

In the *Target* task environment, coordination amongst agents was required only for collision avoidance (both with other agents and obstacles) since the goal positions for all the agents were predetermined. For the *Coverage*, *Formation* and *Line* tasks, the agents not only need to coordinate for collision avoidance but also need to develop consensus on their goals. Table 3 shows the success rate and the fraction of episode taken to complete the task in the *Coverage*, *Formation* and *Line* environments. Here, InforMARL was trained in the 3-agent scenario and tested on the 3- and 7-agent scenarios, whereas RMAPPO is trained and tested on scenarios with the same number of agents. InforMARL is able to achieve a success rate of almost 100% across all scenarios in the different environments, while taking a similar fraction of the episode to complete as RMAPPO. While RMAPPO requires global information to learn a successful policy, InforMARL only needs local neighborhood information. This illustrates the effectiveness of the information aggregation module in InforMARL when the agents only have access to local information.

Environment	m	Metric	Algorithm	
			RMAPPO	InforMARL
Coverage	3	T	0.34	0.36
		$S\%$	100	100
	7	T	0.42	0.43
		$S\%$	100	99
Formation	3	T	0.31	0.30
		$S\%$	100	100
	7	T	0.47	0.43
		$S\%$	100	100
Line	3	T	0.24	0.21
		$S\%$	100	100
	7	T	0.38	0.36
		$S\%$	100	100

Table 3. Performance of RMAPPO and InforMARL on the *coverage*, *formation*, and *line* tasks. We note that InforMARL was trained on the 3-agent scenario and tested on $m = \{3, 7\}$ agents, while RMAPPO was trained and tested on the same number of agents (i.e., with $m = n$).

5. Conclusions and Future Work

We introduced InforMARL, a novel architecture that uses GNNs for scalable multi-agent reinforcement learning. We showed that having just local observations as states is not enough for standard MARL algorithms to learn meaningful policies. Along with this, we also showed that albeit naïvely concatenating state information about all the entities in the environment helps to learn good policies, they are not transferable to other scenarios with a different number of entities than what it was trained on. InforMARL is able to learn transferable policies using standard MARL algorithms using just local observations and an aggregated neighborhood information vector. Furthermore, it has better sample complexity than other standard MARL algorithms that use global observation. We demonstrated these findings for four environments with different navigation tasks: *target*, *coverage*, *formation*, and *line*. Future work will include the introduction of more complex (potentially adversarial)

dynamic obstacles in the environment and adding a safety guarantee layer for the actions of the agents to avoid collisions. Additionally, the use of InforMARL for curriculum learning and transfer learning to different environments is a topic of ongoing research.

Acknowledgements

The authors would like to thank the MIT SuperCloud (Reuther et al., 2018) and the Lincoln Laboratory Supercomputing Center for providing high performance computing resources that have contributed to the research results reported within this paper. The NASA University Leadership initiative (grant #80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity. This research was sponsored in part by the United States AFRL and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notion herein.

References

Ackermann, J., Gabler, V., Osa, T., and Sugiyama, M. Reducing overestimation bias in multi-agent domains using double centralized critics. *CoRR*, abs/1910.01465, 2019. URL http://arxiv.org/abs/1910.01465.

Agarwal, A., Kumar, S., and Sycara, K. P. Learning transferable cooperative behavior in multi-agent teams. *CoRR*, abs/1906.01202, 2019. URL http://arxiv.org/abs/1906.01202.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

Boldrer, M., Palopoli, L., and Fontanelli, D. Socially-aware multi-agent velocity obstacle based navigation for nonholonomic vehicles. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 18–25, 2020. doi: 10.1109/COMPSAC48688.2020.00012.

Chen, S., Dong, J., Ha, P. Y. J., Li, Y., and Labi, S. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Comput.-Aided Civ. Infrastruct. Eng.*, 36(7): 838–857, jun 2021. ISSN 1093-9687. doi: 10.1111/mice.12702. URL https://doi.org/10.1111/mice.12702.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL http://arxiv.org/abs/1409.1259.

Dames, P., Tokekar, P., and Kumar, V. Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, 36(13-14):1540–1553, 2017. doi: 10.1177/0278364917709507. URL https://doi.org/10.1177/0278364917709507.

Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M. G., and Pineau, J. Tarmac: Targeted multi-agent communication. *CoRR*, abs/1810.11187, 2018. URL http://arxiv.org/abs/1810.11187.

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017. doi: 10.1109/TSMC.2016.2582745.

Dunbabin, M. and Marques, L. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1):24–39, 2012. doi: 10.1109/MRA.2011.2181683.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699, 2020. URL https://arxiv.org/abs/2012.09699.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212.

Gronauer, S. and Diepold, K. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

Hausknecht, M. J. and Stone, P. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015. URL http://arxiv.org/abs/1507.06527.

Heess, N., Wayne, G., Silver, D., Lillicrap, T. P., Tassa, Y., and Erez, T. Learning continuous control policies by

stochastic value gradients. *CoRR*, abs/1510.09142, 2015. URL http://arxiv.org/abs/1510.09142.

Hii, M. S. Y., Courtney, P., and Royall, P. G. An evaluation of the delivery of medicines using drones. *Drones*, 3(3), 2019. ISSN 2504-446X. doi: 10.3390/drones3030052. URL https://www.mdpi.com/2504-446X/3/3/52.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Hoshen, Y. Vain: Attentional multi-agent predictive modeling. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/748ba69d3e8d1af87f84fee909eef339-Paper.pdf.

Jiang, J. and Lu, Z. Learning attentional communication for multi-agent cooperation. *CoRR*, abs/1805.07733, 2018. URL http://arxiv.org/abs/1805.07733.

Jiang, J., Dun, C., Huang, T., and Lu, Z. Graph convolutional reinforcement learning. In *ICLR*, 2020.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Khan, A., Tolstaya, E., Ribeiro, A., and Kumar, V. Graph policy gradients for large scale robot control, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. In Solla, S., Leen, T., and Müller, K. (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Kumar, V., Rus, D., and Singh, S. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, 3 (4):24–33, 2004. doi: 10.1109/MPRV.2004.17.

Lazaric, A., Restelli, M., and Bonarini, A. Transfer of samples in batch reinforcement learning. pp. 544–551, 01 2008. doi: 10.1145/1390156.1390225.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015. URL http://arxiv.org/abs/1504.00702.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., and Gao, Y. Multi-agent game abstraction via graph attention neural network. *CoRR*, abs/1911.10715, 2019. URL http://arxiv.org/abs/1911.10715.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017a. URL http://arxiv.org/abs/1706.02275.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017b.

Luders, B., Aoude, G., Joseph, J., Roy, N., and How, J. Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns. 07 2011.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL http://arxiv.org/abs/1310.4546.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *CoRR*, abs/2003.04960, 2020. URL https://arxiv.org/abs/2003.04960.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319289276.

Phillips, M. and Likhachev, M. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pp. 5628–5635, 2011. doi: 10.1109/ICRA.2011.5980306.

Puterman, M. L. *Markov Decision Processes*. Wiley, 1994. ISBN 978-0471727828. URL http://books.google.com/books/about/Markov_decision_processes.html?id=Y-gmAQAAIAAJ.

Qin, Z., Zhang, K., Chen, Y., Chen, J., and Fan, C. Learning safe multi-agent control with decentralized neural barrier certificates. *CoRR*, abs/2101.05436, 2021. URL https://arxiv.org/abs/2101.05436.

Rao, V. and Bernstein, D. Naive control of the double integrator. *IEEE Control Systems Magazine*, 21(5):86–97, 2001. doi: 10.1109/37.954521.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J. N., and Whiteson, S. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. *CoRR*, abs/1803.11485, 2018. URL http://arxiv.org/abs/1803.11485.

Rendleman, J. and Mountin, S. *Responsible SSA Cooperation to Mitigate On-orbit Space Debris Risks*, pp. 851–856. International Conference on Recent Advances in Space Technologies, 2015.

Reuther, A., Kepner, J., Byun, C., Samsi, S., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Jones, M., Klein, A., Milechin, L., Mullen, J., Prout, A., Rosa, A., Yee, C., and Michaleas, P. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6. IEEE, 2018.

Shi, Y., Huang, Z., Wang, W., Zhong, H., Feng, S., and Sun, Y. Masked label prediction: Unified massage passing model for semi-supervised classification. *CoRR*, abs/2009.03509, 2020. URL https://arxiv.org/abs/2009.03509.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., and et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. URL http://arxiv.org/abs/1712.01815.

Sukhbaatar, S., Szlam, A., and Fergus, R. Learning multiagent communication with backpropagation. *CoRR*, abs/1605.07736, 2016. URL http://arxiv.org/abs/1605.07736.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V. F., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning. *CoRR*, abs/1706.05296, 2017. URL http://arxiv.org/abs/1706.05296.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S., Leen, T., and Müller, K. (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *CoRR*, abs/1511.08779, 2015. URL http://arxiv.org/abs/1511.08779.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337. Morgan Kaufmann, 1993a.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993b.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009. URL http://jmlr.org/papers/v10/taylor09a.html.

Tokekar, P., Isler, V., and Franchi, A. Multi-target visual tracking with aerial robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3067–3072, 2014. doi: 10.1109/IROS.2014.6942986.

Turpin, M., Michael, N., and Kumar, V. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33(1):143 – 156, August 2012a.

Turpin, M., Michael, N., and Kumar, V. Decentralized formation control with variable shapes for aerial robots. In *2012 IEEE International Conference on Robotics and Automation*, pp. 23–30, 2012b. doi: 10.1109/ICRA.2012.6225196.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019.

Weeden, C., Goff, J., Noble, J., Schiel, J., Turse, D., Doughan, C., Carrico, J., and Patterson, R. Comments of global newspace operators, 2019. URL https://www.fcc.gov/ecfs/search/search-filings/filing/1040578949828.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=YVXaxB6L2Pl.

Yun, W. J., Lim, B., Jung, S., Ko, Y., Park, J., Kim, J., and Bennis, M. Attention-based reinforcement learning for real-time UAV semantic communication. *CoRR*, abs/2105.10716, 2021. URL https://arxiv.org/abs/2105.10716.

Zhang, H., Wei, S., Yu, W., Blasch, E., Chen, G., Shen, D., and Pham, K. Scheduling methods for unmanned aerial vehicle based delivery systems. In *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, pp. 6C1–1–6C1–9, 2014. doi: 10.1109/DASC.2014.6979499.

Zhou, L., Sharma, V. D., Li, Q., Prorok, A., Ribeiro, A., and Kumar, V. Graph neural networks for decentralized multi-robot submodular action selection. *CoRR*, abs/2105.08601, 2021. URL https://arxiv.org/abs/2105.08601.

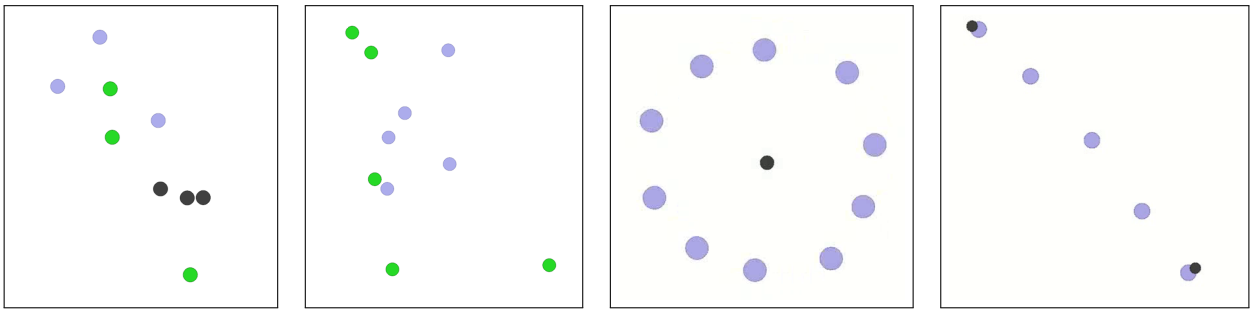
Ángel Madridano, Al-Kaff, A., Martín, D., and de la Escalera, A. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 173:114660, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.114660. URL https://www.sciencedirect.com/science/article/pii/S0957417421001019.

A. Baseline Implementation Sources

We modified the codebases from the official implementations for the GPG, DGN, EMP, and MAPPO baselines and a thoroughly benchmarked codebase for MADDPG, MATD3, VDN and QMIX and provide the links to those implementations here. Note that we used the same hyperparameters as used in their original implementations assuming that they were optimal. We performed a hyperparameter search for these algorithms by varying the learning-rates, network size and a few algorithm specific parameters but did not find a better set of hyperparameters for the environment and chose to report with the original hyperparameters.

- GPG: https://github.com/arbaazkhan2/gpg_labeled
- DGN: https://github.com/jiechuanjiang/pytorch_DGN
- EMP: https://github.com/sumitsk/marl_transfer
- MAPPO: https://github.com/marlbenchmark/on-policy
- MADDPG, MATD3, QMIX, VDN: https://github.com/marlbenchmark/off-policy

B. Environment Tasks



(a) Target (b) Coverage (c) Formation (d) Line

Figure 4. The agents are shown in blue circles, the goals are shown in green and obstacles are shown in black in the Target and Coverage environment. The landmarks are shown in black in the Formation and Line environments.

B.1. Target

There are N agents, N goals along with static obstacles in the environment. Each agent is supposed to go to its distinct goal while avoiding collisions with other entities in the environment. Agents start at random locations at the beginning of each episode; the corresponding goals are also randomly distributed. Each agent i gets a reward: $r_t^{(i)} = r_{\text{dist},t}^{(i)} + r_{\text{coll},t}^{(i)} + r_{\text{goal},t}^{(i)}$, where $r_{\text{dist},t}^{(i)}$ is the negative of the Euclidean distance to the goal, $r_{\text{coll},t}^{(i)} = -5$ if it collides with any other entity and zero otherwise, and $r_{\text{goal},t}^{(i)} = +5$ if the agent has reached the goal and zero otherwise. The joint reward function is defined as $R(s_t, A_t) = \sum_{i=1}^{N} r_t^{(i)}$, which encourages cooperation among all agents.

B.2. Coverage

In the *Coverage* environment (Tokekar et al., 2014; Dames et al., 2017), there are N agents and N goals in the environment. A major difference compared to the *Target* environment is that each agent can go to any goal instead of going to a specific goal. The agents have to avoid collisions with other entities in the environment, and ensure that only one agent is present at each goal. To get the rewards for each agent, a linear sum assignment problem is solved at each time step where agents are assigned to goals depending on which one is the closest. The joint reward for all the agents is the negative of the mean of the minimum Euclidean distances to these assigned goals.

B.3. Formation

In the *Formation* environment (Agarwal et al., 2019), there is a single landmark along with N agents. The agents have to position themselves in an N-sided regular polygon with the landmark at its centre. The agents are rewarded at each time step according to how close they are to their expected positions. These expected positions are assigned by solving a linear sum assignment problem at each time step and depends on the number of agents in the environment and the desired radius of the polygon. We set the target radius to 0.5 for our experiments.

B.4. Line

In the *Line* environment, there are N agents and two landmarks. The agents have to position themselves in an equally spread-out line between these two landmarks. Similar to the *Formation* environment, the agents are rewarded according to how close they are to their expected positions. These expected positions are assigned by solving a linear sum assignment problem at each time step.

C. Hyperparameters

Tables 4, 5, 6, 7 show the hyperparameters for InforMARL, MAPPO, MADDPG, MATD3, QMIX and VDN.

hyperparameters	Value
entity embedding layer dim	3
entity hidden dim	16
num embedding layer	1
add self loop	False
gnn layer hidden dim	16
num gnn heads	3
num gnn layers	2
gnn activation	ReLU

Table 4. Hyperparameters used in InforMARL

Common Hyperparameters	Value
recurrent data chunk length	10
gradient clip norm	10.0
gae lambda	0.95
gamma	0.99
value loss	huber loss
huber delta	10.0
batch size	num envs $\times$ buffer length $\times$ num agents
mini batch size	batch size / mini-batch
optimizer	Adam
optimizer epsilon	1e-5
weight decay	0
network initialisation	Orthogonal
use reward normalisation	True
use feature normalisation	True

Table 5. Common Hyperparameters used in MAPPO and InforMARL

D. Full Comparison

We showcase the learning curves of all the baseline algorithms in this section in Figure 5.

Common Hyperparameters	Value
gradient clip norm	10.0
random episodes	5
epsilon	$1.0 \rightarrow 0.05$
epsilon anneal time	50000 timesteps
train interval	1 episode
gamma	0.99
critic loss	mse loss
buffer size	5000 episodes
batch size	32 episodes
optimizer	Adam
optimizer eps	1e-5
weight decay	0
network initialisation	Orthogonal
use reward normalisation	True
use feature normalisation	True

Table 6. Common Hyperparameters used in MADDPG, MATD3, QMIX, VDN

Common Hyperparameters	Value
num envs	128
buffer length	25
num GRU layers	1
RNN hidden state dim	64
fc layer hidden dim	64
num fc	2
num fc after	1

Table 7. Common Hyperparameters used in MAPPO, MADDPG, MATD3, QMIX, VDN and InforMARL

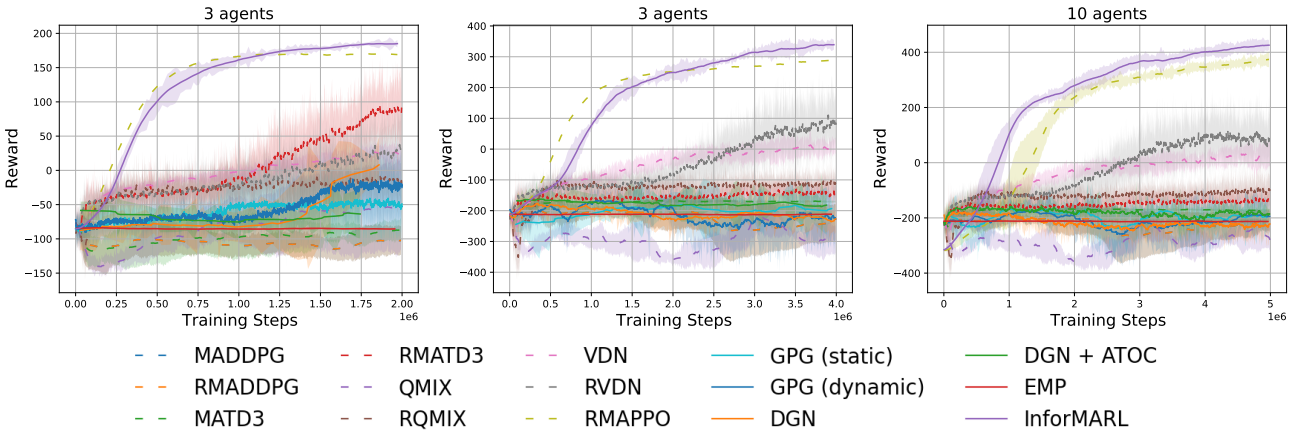


Figure 5. Comparison of InforMARL with baselines. The algorithms that use global information modes are represented with dashed lines and the algorithms that use local information modes are represented with solid lines.

Algorithm	Info mode	$N = 3$				$N = 7$				$N = 10$			
		Reward	T	# col	$S\%$	Reward	T	# col	$S\%$	Reward	T	# col	$S\%$
MADDPG	Global	-100.73	0.97	1.60	5	-206.07	0.98	6.01	0	-210.43	1.00	9.26	0
RMADDPG	Global	10.73	0.75	1.72	23	-122.07	0.95	4.61	3	-127.98	1.00	7.39	0
MATD3	Global	-90.31	0.98	1.11	5	-169.08	0.99	1.98	1	-173.20	1.00	3.50	0
RMATD3	Global	105.49	0.51	1.62	67	-128.93	0.96	3.67	6	-131.72	0.99	5.94	1
QMIX	Global	-54.24	0.84	0.71	7	-288.81	0.97	3.92	0	-273.46	1.00	5.98	0
RQMIX	Global	19.21	0.77	0.83	28	-83.41	0.85	5.83	12	-76.98	0.96	8.65	2
VDN	Global	18.86	0.67	1.56	27	39.87	0.64	4.62	23	43.23	0.73	5.79	19
RVDN	Global	64.04	0.62	0.57	45	140.94	0.62	3.42	47	157.63	0.64	4.93	43
RMAPPO	**Global**	**173.13**	**0.41**	**0.67**	**96**	**327.39**	**0.44**	**4.29**	**88**	**366.81**	**0.44**	**6.93**	**79**
GPG (static)	Local	-67.03	0.96	0.72	7	-180.14	0.99	3.27	1	-182.57	1.00	4.28	0
GPG (dyn.)	Local	-46.27	0.87	0.21	8	-165.91	1.00	1.57	3	-173.53	1.00	2.24	0
DGN	Local	32.94	0.59	1.47	32	-232.32	0.97	2.12	0	-243.45	1.00	4.19	0
DGN+ATOC	Local	67.70	0.66	0.72	35	-189.61	0.97	1.03	0	-201.01	1.00	1.97	0
EMP	Local	-83.96	0.98	0.72	6	-211.90	0.98	2.63	0	-209.90	1.00	94.12	0
InforMARL	**Local**	**205.24**	**0.17**	**1.45**	**100**	**399.01**	**0.37**	**3.72**	**100**	**429.14**	**0.39**	**4.73**	**100**

Table 8. Comparison of InforMARL with other baseline methods, for scenarios with 3, 7, and 10 agents in the environment. The results presented represent the average of 100 test episodes. The following metrics are compared: (a) Total reward obtained in an episode by all the agents (higher is better). (b) Fraction of episode taken by the agents to reach the goal, T (lower is better). (c) The total number of collisions the agents had in the episode, # col (lower is better). (d) Percent of episodes in which all agents are able to get to their goals, $S\%$ (higher is better). The best-performing methods that use global information (RMAPPO) and local information (InforMARL) are highlighted. As noted in Section 4.2, the metrics # col and S should be considered on balance.

E. Ablation Studies

E.1. Graph Information Aggregation Module

The graph information aggregation module allows our method to perform transfer learning to more complex environments. In this section, we compare models with and without the graph information aggregation module. In the absence of the graph information aggregation module, the states of individual agents are concatenated to be given as input to the centralized critic. Figure 6 presents the results of this ablation study. We find that both models have similar sample complexities and performance. However, the number of parameters for the critic network with the graph information aggregation module is much smaller and independent of the number of entities in the environment.

E.2. Effect of Sensing Radius

We investigate how the performance of InforMARL depends on the sensing radius, namely, how much information (i.e., over what neighborhood of the ego-agent) an agent has access to. As the radius increases to a large value, the graph becomes fully connected; as the radius decreases to zero, the neighborhood information mode converges to the local information mode. As seen in Figure 7, when learning with a small sensing radius (*e.g.* $\rho = \{0.1, 0.2\}$), the agents are not able to achieve the same reward as can be achieved with a larger sensing radius ($\rho = \{0.5, 1, 2, 5\}$)[1]. We also note that there are diminishing returns when increasing the sensing radius from $\rho = 0.5$ to $\rho = 5$. Physically, a radius of $\rho = 0.5$ is slightly more than double the distance an agent can traverse in the next two time steps, whereas $\rho = 0.2$ is the distance it can travel in the current timestep. Since the agents far away from the ego-agent have little influence on their immediate decisions, the extra information obtained by increasing the sensing radius does not improve performance very much.

[1]Measurements for ρ, a distance, are in meters

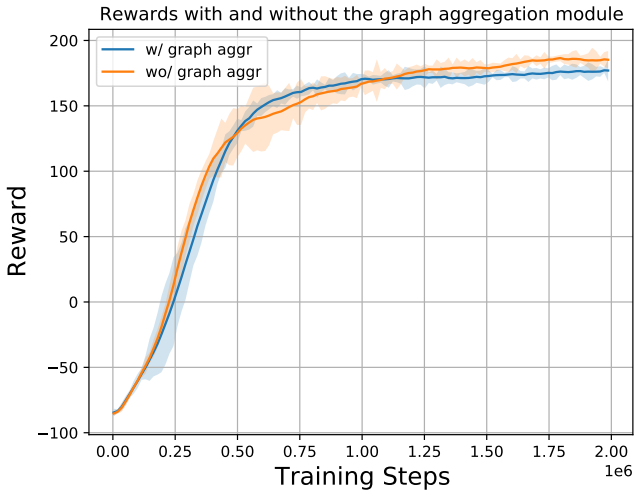


Figure 6. Training performance of InforMARL with, and without, the graph information aggregation module, for a 3-agent scenario. The two variants have similar sample complexities. However, the critic network with the graph information aggregation module has fewer parameters than the one without this module.

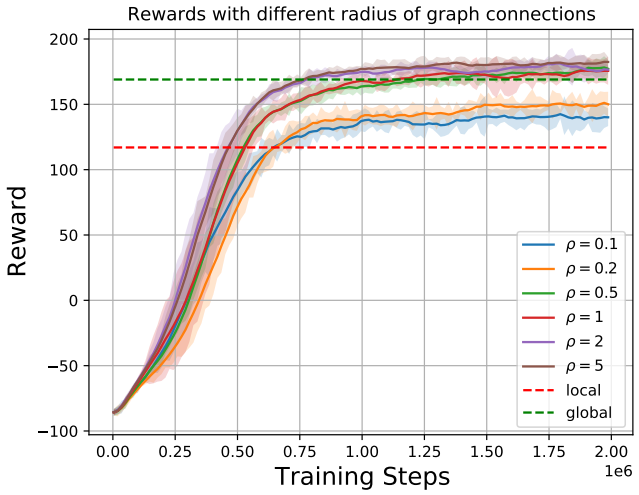


Figure 7. Diminishing returns in performance gains from increasing the sensing radius for InforMARL. The dashed lines are the reward values after saturation for RMAPPO in the global (in green) and local (in red) information modes, respectively. They are provided for reference.