

# Decentralized Multi-Agent Reinforcement Learning with Global State Prediction

Joshua Bloom, Pranjal Paliwal, Apratim Mukherjee, and Carlo Pinciroli

**Abstract**—Deep reinforcement learning (DRL) has seen remarkable success in the control of single robots. However, applying DRL to robot swarms presents significant challenges. A critical challenge is non-stationarity, which occurs when two or more robots update individual or shared policies concurrently, thereby engaging in an interdependent training process with no guarantees of convergence. Circumventing non-stationarity typically involves training the robots with global information about other agents’ states and/or actions. In contrast, in this paper we explore how to remove the need for global information. We pose our problem as a Partially Observable Markov Decision Process, due to the absence of global knowledge on other agents. Using collective transport as a testbed scenario, we study two approaches to multi-agent training. In the first, the robots exchange no messages, and are trained to rely on implicit communication through push-and-pull on the object to transport. In the second approach, we introduce Global State Prediction (GSP), a network trained to form a belief over the swarm as a whole and predict its future states. We provide a comprehensive study over four well-known deep reinforcement learning algorithms in environments with obstacles, measuring performance as the successful transport of the object to a goal location within a desired time-frame. Through an ablation study, we show that including GSP boosts performance and increases robustness when compared with methods that use global knowledge.

## I. INTRODUCTION

Reinforcement learning (RL) has shown promise in the control of single agents. However, coordinating multi-agent systems through RL (MARL) still poses significant challenges. Among them, a critical one is *non-stationarity*, which occurs when independently learning agents change their policy during training. This breaks the Markov assumption, which in turn removes convergence guarantees due to the likelihood of agents entering an infinite loop of mutual adaptation around sub-optimal policies.

Non-stationarity is typically circumvented through diverse techniques, which include the use of a centralized policy controlling all agents simultaneously, introducing global state or action information during training, or performing complex message passing [1], [2], [3], [4], [5].

In this paper, we explore how to circumvent non-stationarity without resorting to the agents having explicit knowledge of each other’s actions and states. We analyze two approaches: one in which the agents do not exchange messages, and one in which the agents exchange minimal, partial, and local information and learn to build a prediction of the global state.

All authors are with Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA, USA (email: {jdbloom, ppaliwal, amukherjee, cpinciroli}@wpi.edu).

We showcase our approach in a collective transport scenario, in which a team of robots is physically connected around an object that must reach a predefined location. This scenario is representative of the larger category of *robotic aggregates*, i.e., teams of robots connected to each other to form a rigid lattice. Modular robots [6], self-assembling robots [7], swarm formation control [8], and certain approaches to collective transport [9] fall into this category. Controlling aggregates is in itself an open research question for MARL due to additional challenges such as continuous control and partial observability of the global state.

In our experimental scenario, implicit (i.e., message-free) communication (IC) occurs through the natural push-and-pull that the robots experience by moving the object. We show that this form of communication is sufficient for the robots to learn a control policy capable of emergent coordination.

In a second set of experiments, we introduce Global State Prediction (GSP). GSP is a neural network that collects partial local observations and predicts future state changes at the team level. This prediction is then fed as an additional input to the control policy network. GSP removes the need for explicit global information, while imposing minimal communication requirements.

To train these approaches, we propose Aggregate Centralized Training with Decentralized Execution (A-CTDE). Through this technique, robots store their local experiences in a shared memory for single policy training. This policy is then placed on all robots for decentralized execution. Unlike Centralized Training with Decentralized Execution, all stored local experiences have been directly influenced by the other agents and not simply treated as part of the environment.

We compare IC and GSP against a previously proposed method where robots share Global Knowledge (GK) via direct broadcast communication [10]. We study the performance of four well-known training methods: DQN, DDQN, DDPG, and TD3. Our results indicate that robots operating only with IC are able to complete the task with relatively high success. When GSP is added, our results out-perform IC and even GK. GSP has lower communication requirements than GK, and also eliminates the need for global knowledge. We present an in-depth analysis on the behaviors across all three methods and explore the effect each robots’ shared information has on GSP.

## II. BACKGROUND AND RELATED WORK

### A. Deep Reinforcement Learning

Reinforcement Learning (RL) is typically formalized as a Markov Decision Process  $\langle S, A, R, T \rangle$ , where  $S$  is the set

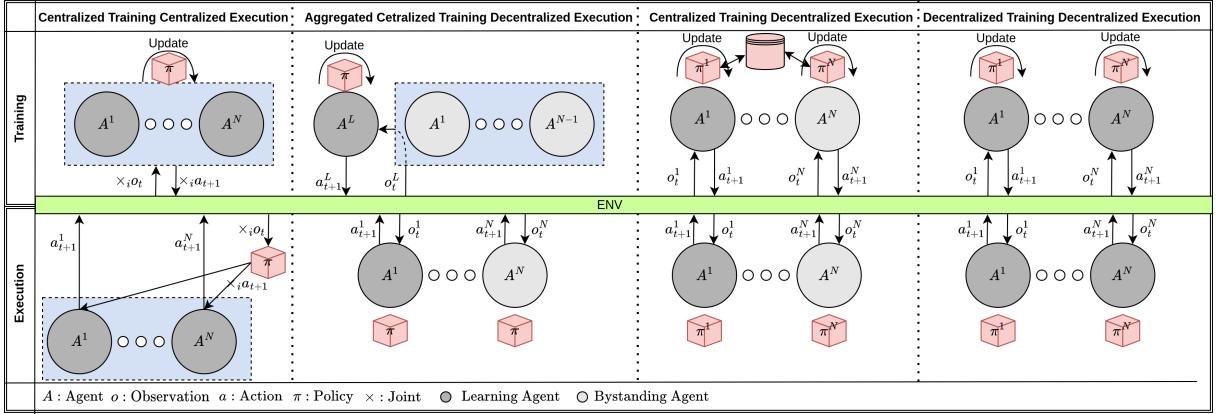


Fig. 1. Common schemes for training and executing Deep Reinforcement Learning Algorithms for Multi-Agent environments. CTCE trains a single policy on global information and produces a joint action. A-CTDE trains a single policy on local information and single agent actions, is then copied and placed on each actor for local execution. CTDE trains multiple policies on local information with the addition of global information, the global information is removed during execution. DTDE trains agent policies independently on local information and executes independently on local information

of all possible discrete states,  $A$  is the set of all available discrete actions,  $R(s, a, s') : S \times A \times S \rightarrow \mathbb{R}$  is a reward function, and  $T(s'|s, a)$  is a probabilistic transition function mapping states and actions to new states. Optimal decision-making in an MDP is captured by the Bellman Equation:

$$Q(s, a) = \sum_{s'} T(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')],$$

where  $Q(s, a)$  is a function mapping state-action pairs to a reward, and  $\gamma$  is a discount factor.

RL has shown success when states and actions are discrete, and when the number of state-action pairs is limited. However, when working with robots, it is often unavoidable to consider continuous state spaces. Deep Reinforcement Learning (DRL) offers a solution by expressing the mapping  $Q(s, a)$  as a function  $Q(s, a|\theta)$  where  $\theta$  is a set of neural network parameters.

Using DRL in realistic environments often presents the issue of noisy or incomplete observations. This is typically captured as a Partially Observable Markov Decision Process  $\langle S, A, R, T, \Omega, O \rangle$ , where  $o \in \Omega(s)$  is a partial observation of the full state  $s \in S$  according to a probabilistic function  $O(o|s)$ .

Here we employ four well known DRL algorithms: Deep Q Network (DQN) [11], Double DQN (DDQN) [12], Deep Deterministic Policy Gradient (DDPG) [13], and Twin Delayed DDPG (TD3) [14]. We chose these four DRL implementations due to their recent advancement in current robotics research [15].

Note the difference in control. DQN and DDQN approximate the  $Q(s, a)$  and, as such, return expected future values for each possible action given a state. An  $\epsilon$ -greedy training schema is then employed where, with probability  $(1 - \epsilon)$ , the agent chooses greedily over the action space, and with probability  $\epsilon$  it chooses randomly over the action space. Parameter  $\epsilon$  is decayed until reaching a minimum set value. DQN and DDQN require a discrete action space. Alternatively, DDPG and TD3 directly output the actions to

take, or the *policy*, as a real-valued number. Therefore, the action space is continuous.

### B. Multi-Agent Reinforcement Learning

Moving from a single agent to a multi-agent setting introduces several complexities in learning [16]. There are various approaches to tackling multi-agent scenarios, typically categorized in terms of distributedness across two axes: training and execution [1]. As shown in Figure 1(A), Centralized Training with Centralized Execution (CTCE) employs a single trained policy that observes a global joint observation space and selects a joint action for all agents to execute. CTCE is a natural extension for DRL methods to multi-agent scenarios. However, the state-action space grows exponentially with the number of agents.

In contrast, Decentralized Training with Decentralized Execution (DTDE), shown in Figure 1(D) trains every agent independently, treating all other agents as part of the environment. Ideally, this learning schema is desirable because it theoretically splits a large, global training problem into more manageable local instances. However, this approach does not guarantee convergence because of non-stationarity, which breaks the Markov assumption.

To cope with non-stationarity, one can combine the benefits of a centralized trainer with the flexibility of decentralized execution (CTDE), as shown in Figure 1(C). In this approach, each agent trains its own local policy with the addition of external global information only available during training. This external information is then removed during execution. Foerster *et al.* [3] introduce learners who communicate (CTDE), allowing for differentiation, via discretized/regularized units (DRU), between agents during training. DRUs provide a way to directly calculate the gradient corresponding to how well other agents received messages. While CTDE methods like MADDPG [17], A3C [18], MATD3 [19], PPO [20], VDN [21], and Q-Mix [22] have shown remarkable success in MARL when they know the positions of all agents involved, they fail to learn when

the information shared is restricted to just the local information [23]. Our work seeks to leverage the benefits of developing a single policy shared amongst all of the agents, such as resilience to attrition and repeatability given identical observations while operating on local information only.

A recent, alternative idea for dealing with non-stationarity in DTDE or CTDE is to locally model individual agents and produce a belief over their future actions [5], [24]. Wu *et. al* [4] introduce *spatial intention maps* which combine with the agent's observation space to provide spatial information about other agent's actions. Agents in this environment communicate their belief over other agents' actions via a set of waypoints indicating predicted trajectories. The agents compile these communicated trajectories and build a global map of beliefs. Other work incorporating belief in the multi-agent setting includes reasoning directly what your own policy would have you do given another agents' observations [25], [26], and rewarding causal influence over other agents' actions [27], [28].

### C. Our Contribution

While powerful, the mentioned methods based on belief are limited in their prediction abilities, because reasoning over agents individually is prone to scaling problems. In this work, we propose a new approach to modeling belief over agents in a MARL setting called Global State Prediction (GSP). GSP is rooted in the human ability to psychologically represent others, often referred to as *Theory of Mind* [29]. However, rather than forming a belief over each agent individually, GSP abstracts further, similar to *herd mentality* [30]. GSP is specifically trained to predict the outcome of all agents' actions given a limited amount of local partial information from each agent. Unlike auxiliary tasks [31], [32], [33], [34], [35], [36], GSP is a separate network whose output is used only as an observation by each agents' policy to aid with action selection.

## III. METHODOLOGY

GSP works by collecting a subspace of each robots' observation space  $\vec{I} \in O$  producing a prediction on the global state of the robot aggregate in the future. Unlike Zhang *et. al* [10], our method does not require full state space information. Rather, it is sufficient to share aggregate information, e.g., the average of each robots' proximity readings. This input is both less informative and more realistic to retrieve and communicate than global knowledge as defined in [10]. GSP then takes these averaged readings and produces a prediction over the change in orientation of the robot aggregate. This prediction is appended to the local observation spaces of each robot for action selection as shown in Figure 2.

### A. Aggregate Centralized Training Decentralized Execution

We implement a version of Centralized Training with Decentralized Execution (CTDE) similar to Radulescu *et al.* [37], however, our method differs due to the physical connections between learning agents. This connection allows agents to mutually affect each others' states after action

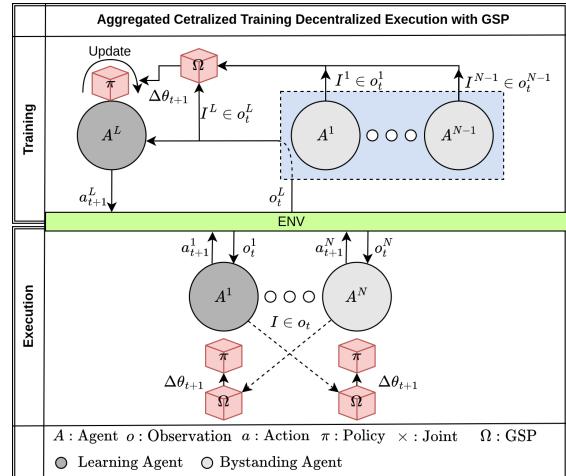


Fig. 2. GSP implementation with A-CTDE

execution. In this implementation, all of the agents store their local experiences in a shared replay, from which a single policy is learned. This allows us to bootstrap the policy, and because we are learning from local experiences without global knowledge, this local policy can be scaled after training to more or less agents than was trained. We will refer to this version as Aggregate Centralized Training Decentralized Execution (A-CTDE), as shown in Figure 1(B).

During training, everything is centralized and the policy is queried once per agent per time-step, taking as input the local observation of that agent and producing an action for that agent to take. Once all actions are compiled, the agents execute their respective actions. At execution, this single policy is copied and placed on each agent individually for that agent to run locally and asynchronously.

Although this method of training allows us to treat training as a single agent, it is still susceptible to environmental non-stationarity because, even though all the agents are running the same policy, they are concurrently updating this shared policy [37]. From the perspective of a single agent, all of the other agent's policies are changing through time.

### B. Addressing Environmental Non-Stationarity in Robot Aggregates

Environmental non-stationarity of a single agent  $i$  can be modeled in the transition probability function  $T(s'_i|s_i, a_i, \rho_{\pi(\theta)})$ , where  $\rho_{\pi(\theta)}$  is the transition dynamics following a hidden and changing distribution defined by the current parameters  $\theta$  of the policy  $\pi$ .

This hidden and changing distribution  $\rho$  is the direct cause of environmental non-stationarity. If there was only one learning agent present in the environment, then  $\rho$  would be hidden but not changing and, as such, would be learnable. But, because all of the agents are updating their shared policy concurrently, the hidden distribution defining  $\rho$  is also moving, thus leading to the moving target problem [1].

We explore dealing with environmental non-stationarity in three different ways: implicit communication (IC), globally shared knowledge (GK), and global state prediction (GSP).

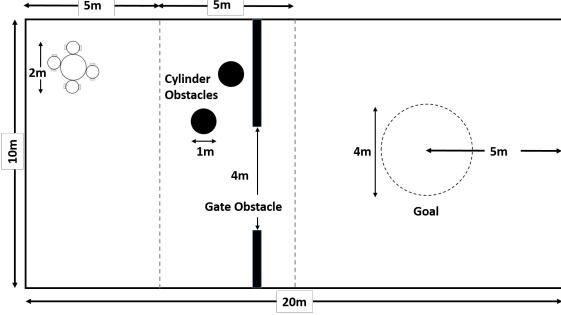


Fig. 3. Evaluation environment for a collective transport task with obstacles present. The left-most region represents the robot aggregate generation zone, the center region represents the obstacle generation zone, and the region on the right represents the goal

1) *Implicit Communication*: By the definition of a robot aggregate in this paper, all of the robots are rigidly attached in some pre-defined lattice, naturally inducing implicit communication via push-and-pull on their connection to the other robots. This push-and-pull is observed by all of the other agents in the error they experience when they execute a selected action. As shown in Sec. IV, this form of implicit communication proves to be sufficient for DRL to adequately model  $\rho_{\pi(\theta)}$  and produce viable policies.

2) *Global Knowledge*: As a baseline for our work, we consider the use of complete and global knowledge (GK) on the robot actions and states. We implement the method presented by Zhang *et al.* [10], in which the robots broadcast the current global position and observed velocity vectors  $(\vec{x}, \vec{\dot{x}})$  of every robot in the aggregate. The set of all position and observed velocity vectors excluding the observing robot's vectors is  $(X^{-i}, \dot{X}^{-i})$  and is appended to the local observation spaces of every robot  $i$ . By observing  $(X^{-i}, \dot{X}^{-i})$  the changing effect  $\rho$  exhibits as a result of concurrent updates on  $\pi$  is eliminated because the robots are communicating their policy  $\pi(s)$  allowing robot  $i$  to understand the joint action space  $\times A_t$ . Thus, non-stationarity is eliminated, allowing DRL to reach viable policies.

3) *Global State Prediction*: Communicating  $(\vec{x}, \vec{\dot{x}})$  is not always possible or accurate, and may not construe all relevant information to the environment, such as observations from sensors. Furthermore, global information about the robot aggregate, such as positions, may not be available in most realistic scenarios. In keeping with the example presented in [10], GSP receives only  $X^{-i}$  and provides a prediction over the change in orientation of the robot aggregate. Due to the centralized training present in A-CTDE, the actual change in angle is known in the following time-step for GSP to be trained on. The prediction of the change in angle is a prediction of the result over the joint action  $\times A_t$  taken during the current time-step thus eliminating the moving target problem present in  $\rho$  and allowing DRL to construct viable policies.

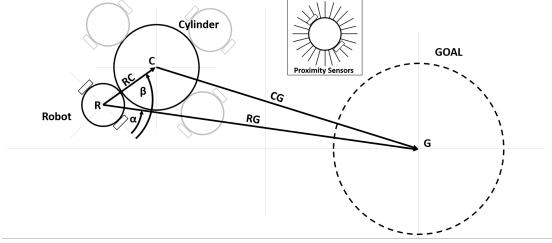


Fig. 4. Robot observation space consisting of the vectors from the robot to the object to transport and to the goal, the distance from the object to transport to the goal, the wheel velocities, and the array of 24 proximity sensors

#### IV. EVALUATION

##### A. Experimental Setup

1) *Collective Transport*: We evaluate all methods, IC, GK, and GSP, on a collective transport task as shown in Figure 3. This task is simulated using the ARGoS multi-robot simulator [38] with the Buzz programming language [39] for individual robot control and the PyTorch library for the Neural Network infrastructure.

The environment is defined such that the robot aggregate will be generated in the left-most region, where the pose of the object to transport is randomly generated according to a uniform distribution. The robots are then randomly placed at equal mutual distances around the object.

The central region of the environment is the *obstacle generation zone*. We evaluate our work against two types of obstacles: cylinder obstacles and a gate obstacle. The positions of the cylinder obstacles are randomly generated according to a uniform distribution. The horizontal position of the gate obstacle and the vertical position of the center of the opening are randomly generated according to a uniform distribution. Both the cylinder obstacles and the gate obstacle are constrained to be within the obstacle generation zone which lies between the *robot aggregate generation zone* and the *goal zone*.

2) *Robot Control and Sensing*: We chose the foot-bot [40] due to its differential-drive controller and independent non-actuated turret attached to an actuated gripper. The turret allows the robot to rotate independently from the gripper. The gripper is not actively controlled and is only actuated upon initialization or failure.

The robots are controllable through wheel velocities  $v \in (-10, 10)$  cm/s. The agent chooses a  $\Delta v \in \{-0.1, 0, 0.1\}$  cm/s in the case of DQN and DDQN, and  $\Delta v \in [-0.1, 0.1]$  cm/s in the case of DDPG and TD3. Note the distinction between a discrete action space and a continuous action space as DQN and DDQN choose their actions via an epsilon-greedy strategy, whereas DDPG and TD3 output their actions directly. In the case of DQN and DDQN, we must provide an action space consisting of all possible combinations of actions between the two wheels, represented as  $|\Delta v|^2$ . We choose a discretization of  $|\Delta v| = 3$  for control simplicity.

All robots in the environment are identical and are able to sense several values as shown in Figure 4. They observe

---

**Algorithm 1** IC

---

```

Initialize Policy  $\pi_0$  and Buffer  $B$ 
for each episode do
    Initialize Robots, Object, and Obstacles
    Receive initial observations  $O_t$ 
    while not done at timestep  $t$  do
        for each robot  $i$  do
             $a_t^i \leftarrow \pi_t(o_t^i)$ 
        end for
        Execute actions  $A_t$  and receive  $(O_{t+1}, R_t, \text{Done})$ 
        for each robot  $i$  do
             $B \leftarrow (o_t^i, a_t^i, r_t^i, o_{t+1}^i, \text{Done})$ 
        end for
         $\pi_{t+1} \leftarrow \text{Update Policy } \pi_t$ 
         $o_t \leftarrow o_{t+1}$ 
    end while
end for

```

---

the vector from themselves to the goal  $\vec{RG}$ , the vector from themselves to the object to transport  $\vec{RC}$ , the distance from the object to transport and the goal  $|CG|$ , their wheel speeds  $v$ , and their 24 proximity sensors  $p$  uniformly distributed around the robot. The proximity sensors can sense a distance of 2 m returning a value of 0 for no observation and 1 for touching the sensed object.

3) *Global State Prediction*: GSP only requires the robots to communicate their average proximity values  $\sum p / |p|$ , a single floating-point value. These values are used as input to GSP, which then produces a predicted change in the orientation of the robot aggregate as a result of the actions executed at the current time-step. This prediction is appended to the observation space of the robots. GSP collects raw sensory information as opposed to GK presented in Zhang *et al.* [10] which uses actual speeds and positions without noise.

4) *Training*: We train policies in environments with either two cylinder obstacles or a gate obstacle. When training with a gate obstacle, we employ curriculum learning [41], with the opening distance of the gate starting equal to the vertical width of the environment (i.e., no obstacle). The opening is then shortened by 0.5 m at a fixed episodic frequency until the opening distance reaches the desired minimum.

The reward function used for all training rewards moving in the direction of the goal while penalizing proximity readings and time taken to complete the task:

$$R_{i,t} = -2 + \frac{\vec{CG} \cdot \left( C(x_t, y_t) - C(x_{t-1}, y_{t-1}) \right)}{|\vec{CG}| \cdot |C(x_t, y_t) - C(x_{t-1}, y_{t-1})|} - \frac{1}{|\mathbf{p}|} \sum \mathbf{p}. \quad (1)$$

Algorithm 1 reports how A-CTDE works with IC. Due to the physical connections in the robot aggregate, we are able to train a single policy, trained on local observations, that is executed on every robot.

We formalize GSP in Algorithm 2. Here, all of the averaged proximity reading  $\mathbf{P}$  are broadcasted and used as input to the GSP network  $\Omega(\mathbf{P}) \rightarrow \Delta\theta$ , where  $\Delta\theta$  is the predicted change in the orientation of the robot aggregate. During training, this prediction is evaluated in the next time-step against the ground truth.

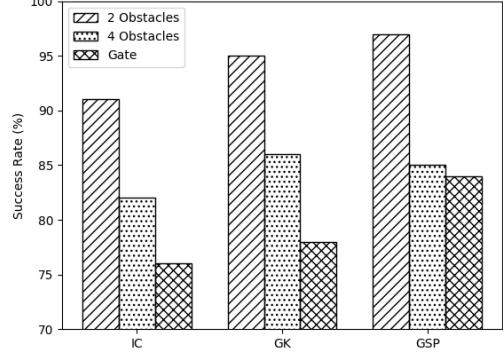


Fig. 5. Comparison of TD3 using Implicit Communication (IC), Global Knowledge (GK), and Global State Prediction (GSP) evaluated on 2 obstacles, 4 obstacles, and the gate obstacle

5) *Neural Network Architectures and Hyperparameters*: All networks architectures are built using three fully connected layers activated using ReLU and optimized using ADAM. DQN and DDQN have size (31, 64, 128, 9) and DDPG and TD3 have size (31, 400, 300, 2). We train GSP using DDPG.

For DQN and DDQN, we drew inspiration from Mnih *et al.* [11] for our choice in hyperparameters. Specifically, a discount factor  $\gamma$  of 0.99997 and a learning rate  $\eta$  of  $10^{-4}$  were used. We used mini-batch learning with a memory of  $10^6$  experiences and a batch size of 100, where an experience is comprised of the initial state, action taken, reward gained, the resultant state, and a boolean terminal flag. We used an  $\epsilon$ -greedy policy with a linear decay of  $10^{-6}$  and a minimum  $\epsilon$  value of 0.01. The target network was updated every 1,000 learning iterations and, during training, learning occurred every time-step. For DDPG and TD3 we set hyperparameter values identical to Lillicrap *et al.* [13]. GSP was trained via DDPG using the same network architecture and hyperparameters.

## B. Experimental Evaluation

1) *Obstacles*: We experiment on 4-robot aggregates in three obstacle environments: 2-cylinder obstacles, 4-cylinder obstacles, and the gate obstacle. To avoid evaluating on training environments, the random seed was changed from training and kept to the same value across all experiments. By keeping the seed the same across testing experiments, we are able to directly compare policies on the same environments.

We compare the successful transport of the object to the goal location within the required time. Each policy was tested in 100 different random orientations of robots, object to transport and obstacles. We report the percentage of successful runs in Table I.

Obstacles provide an interesting evaluation scenario because they provide stimulus to some robots in the aggregate, but not all robots may be able to observe them. Thus, a more sophisticated degree of coordination is required over an environment with no obstacles, where the robots may learn to simply drive in the direction of the goal. We train and test

TABLE I  
SUCCESS RATES WITH OBSTACLES

Obstacles	DQN-IC	DQN-GSP	DDQN-IC	DDQN-GSP	DDPG-IC	DDPG-GSP	TD3-IC	TD3-GSP
2	73%	86%	90%	94%	91%	94%	91%	97%
4	56%	84%	81%	82%	86%	87%	82%	85%
Gate	72%	73%	74%	80%	77%	80%	76%	84%

## Algorithm 2 GSP

```

Initialize Policy  $\pi_0$  and Buffer  $B_1$ 
Initialize GSP  $\Omega$  and Buffer  $B_2$ 
for each episode  $t$  do
    Initialize Robots, Object, and Obstacles
    Receive initial observations  $O_t$ 
    for each robot  $i$  do
        Broadcast  $\mathbf{P}_t^i = \sum \mathbf{p}_t^i / \mathbf{p}_t^i$ 
    end for
     $\Delta\theta_t \leftarrow \Omega(\mathbf{P}_t)$ 
    while not done at timestep  $t$  do
        for each robot  $i$  do
             $a_t^i \leftarrow \pi_t(o_t^i, \Delta\theta_t)$ 
        end for
        Execute actions  $A_t$  and receive  $(O_{t+1}, R_t, R_t^{\Delta\theta}, \text{Done})$ 
        for each robot  $i$  do
            Broadcast  $\mathbf{P}_{t+1}^i = \frac{1}{\mathbf{p}_{t+1}^i} \sum \mathbf{p}_{t+1}^i$ 
        end for
         $\Delta\theta_{t+1} \leftarrow \Omega(\mathbf{P}_{t+1})$ 
        for each robot  $i$  do
             $B_1 \leftarrow ((o_t^i, \Delta\theta_t), a_t^i, r_t^i, (o_{t+1}^i, \Delta\theta_{t+1}), \text{Done})$ 
        end for
         $B_2 \leftarrow (\mathbf{P}_t, \Delta\theta_t, R_t^{\Delta\theta}, \mathbf{P}_{t+1}, \text{Done})$ 
        Update Policy  $\pi$  and GSP  $\Omega$ 
         $o_t \leftarrow o_{t+1}$ ,  $\Delta\theta_t \leftarrow \Delta\theta_{t+1}$ 
    end while
end for

```

on the 2-cylinder obstacle environment as a baseline and then increase the complexity to 4-cylinder obstacles.

The results allow us to draw two conclusions. Firstly, as the complexity of the learning algorithm increases, so does the success rate. This result is in line with previous findings in the literature. Secondly, GSP increases the success rate on all obstacles for all methods. In the case of DQN, the increase is substantial, with success rates on the 4-obstacle environment increasing by 28%. In addition, when we observe the behavior of DRL with GSP, as shown in Figure 6, we notice active obstacle avoidance, depicted by the changing orientation of the robot aggregate shown as the black lines.

The gate obstacle proves to be harder than the 2-obstacle environment. A variant of the bug algorithms is a reasonable behavior for this type of obstacle, where the robot aggregate follows the wall until it finds the opening. However, IC is unable to learn it. We attribute this inability to the need for the robot aggregate to occasionally move *away* from the target to reach the opening in certain environments. However, as shown in Figure 7, GSP learns a successful strategy.

When we evaluate IC and GSP against GK, we observe that the success rates increases in the 2-obstacle environment and the gate environment when using GSP over using GK. The 4-obstacle environment shows similar results. GK performed similarly across all methods, Figure 5 reports the

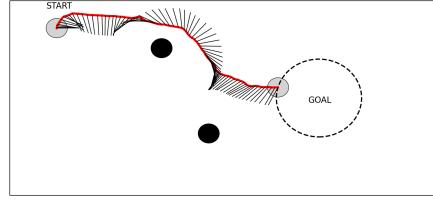


Fig. 6. Trajectory Behavior of DQN with GSP plotted with the orientation of the robot aggregate over the episode

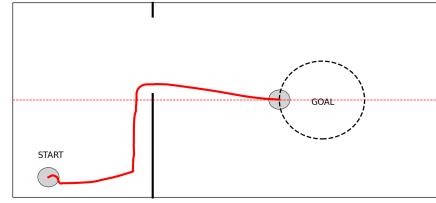


Fig. 7. Trajectory Behavior of DQN with GSP on the gate obstacle with the opening on the opposite side of the environment from the initialization of the robot aggregate

results for TD3. This is an interesting result, considering GSP uses less information than GK, and also forgoes the need for global knowledge.

2) *Resilience*: Resilience is an important feature for multi-robot systems. For robot aggregates, in particular, failures introduce asymmetries in the dynamics. We define a failure as a complete loss of power resulting in a disengagement between the robot and the aggregate. We explore an 8-robot aggregate with a chance of having up to 75% of the robots fail during an episode. During episode initialization, we select the number of robots to fail by giving each robot a 25% chance of failure up to the desired 75% of robot failures. Once the number of failures is determined we randomly assign a failure time.

Table II reports our findings evaluating IC, GK, and GSP on TD3 in 2- and 4-cylinder obstacle environments. In both environments GSP outperforms GK and substantially increases the success rate over IC.

3) *GSP Network Analysis*: To gain insight into the behavior of GSP, we present an analysis of a trained GSP network in a 4-robot aggregate, shown in Figure 8. We simulate the robot aggregate's proximity readings  $\{P_1, \dots,$

TABLE II  
SUCCESS RATES WITH OBSTACLES AND FAILURES

Obstacles	TD3-IC	TD3-GK	TD3-GSP
2	83%	89%	95%
4	67%	79%	81%

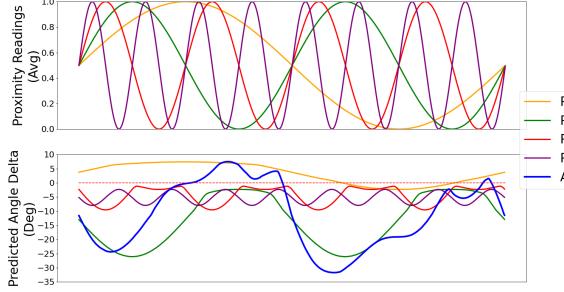


Fig. 8. Top: Simulated averaged proximity values for a 4 robot aggregate; Bottom: GSP network predicted change in angle corresponding to single robot inputs and all inputs combined

P4} communicated across the robots with sine functions over a period of  $2\pi$  with the frequency increasing for each robot. By increasing the frequency, we can better visualize the additional value that input has on the predicted change in orientation angle.

The bottom plot shows the predicted change in orientation angle, where each line is the result of only receiving the corresponding sine function in the top plot. For example, the P1 line shown in orange at the bottom is the result of only receiving the P1 sine wave and keeping other inputs 0.

From this analysis, we notice that all robots play a role in predicting the change in orientation angle of the robot aggregate. However, some robots have a larger impact than others. Robots 0 and 1 are key to generating a larger prediction, with robot 2 playing a moderate role and robot 3 having the smallest impact on the prediction. Upon further investigation, as shown in Figure 9, we found that the robot aggregate reorients at the start of an episode so that robot 1 faces the goal, robots 0 and 2 faces the north and south walls, respectively, and robot 3 is on the backside, opposite of robot 1. This form of role allocation is an emergent property that was not explicitly rewarded during training.

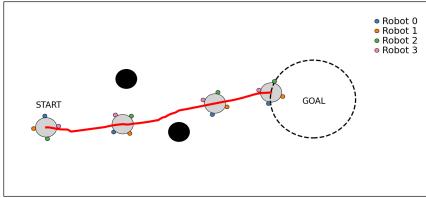


Fig. 9. Emergent reorientation behavior of the robot aggregate with GSP when solving a 2-cylinder obstacle environment

## V. CONCLUSION

We present two approaches to addressing non-stationarity in MARL without the use of global information, evaluated in a collective transport scenario. The first approach is based on implicit communication realized through pushing and pulling the object instead of message passing. The second approach consists of endowing the robots with a network trained to predict the future state of the system (in our case the object) by aggregating partial local observations. We evaluate the performance of our method through four well known reinforcement learning algorithms.

We show that IC is sufficient to achieve coordinated collective transport in environments with obstacles. Furthermore, we demonstrate the ability of GSP to perform better than a prior method that used global knowledge. We provide an in-depth analysis into the mechanisms driving coordination.

We plan on further developing the idea of minimizing non-stationarity through global state prediction from partial observations. In particular, we will apply this concept to tasks beyond collective transport, such as collective motion and multi-robot foraging, and will investigate how more complex MARL methods such as MADDPG [17] and A3C [42] perform along side GSP.

## REFERENCES

- [1] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, Feb. 2022. [Online]. Available: <https://link.springer.com/10.1007/s10462-021-09996-w>
- [2] C. Amato, G. Konidaris, and L. P. Kaelbling, "Modeling and Planning with Macro-Actions in Decentralized POMDPs," *Journal of Artificial Intelligence Research*, vol. 64, pp. 817–859, 2019.
- [3] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to Communicate with Deep Multi-Agent Reinforcement Learning," *Advances in neural information processing systems*, vol. 29, May 2016, arXiv: 1605.06676. [Online]. Available: <http://arxiv.org/abs/1605.06676>
- [4] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial Intention Maps for Multi-Agent Mobile Manipulation," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8749–8756, 2021.
- [5] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning," *arXiv:1906.04737 [cs, stat]*, Jun. 2019, arXiv: 1906.04737. [Online]. Available: <http://arxiv.org/abs/1906.04737>
- [6] Y. Sun, Y. Jiang, H. Yang, L.-C. Walter, J. Santoso, E. H. Skorina, and C. Onal, "Salamanderbot: A soft-rigid composite continuum mobile robot to traverse complex environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2953–2959, iSSN: 2577-087X.
- [7] F. Mondada, G. C. Pettinaro, A. Guignard, I. W. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo, "Swarm-Bot: A New Distributed Robotic Concept," *Autonomous Robots*, vol. 17, no. 2, pp. 193–221, Sep. 2004. [Online]. Available: <https://doi.org/10.1023/B:AURO.0000033972.50769.1c>
- [8] L. Barnes, M. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *2007 Mediterranean Conference on Control & Automation*. Athens, Greece: IEEE, Jun. 2007, pp. 1–8. [Online]. Available: <http://ieeexplore.ieee.org/document/4433724/>
- [9] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, "Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art," *Frontiers in Robotics and AI*, vol. 5, p. 59, May 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/frobt.2018.00059/full>

- [10] L. Zhang, Y. Sun, A. Barth, and O. Ma, "Decentralized Control of Multi-Robot System in Cooperative Object Transportation Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 184 109–184 119, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9201368/>
- [11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: <http://www.nature.com/articles/nature14236>
- [12] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
- [13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, 2015, arXiv: 1509.02971. [Online]. Available: <http://arxiv.org/abs/1509.02971>
- [14] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
- [15] E. F. Morales, R. Murrieta-Cid, I. Becerra, and M. A. Esquivel-Basaldua, "A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning," *Intelligent Service Robotics*, vol. 14, no. 5, pp. 773–805, Nov. 2021. [Online]. Available: <https://link.springer.com/10.1007/s11370-021-00398-z>
- [16] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, Nov. 2019. [Online]. Available: <http://link.springer.com/10.1007/s10458-019-09421-1>
- [17] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *arXiv:1706.02275 [cs]*, Mar. 2020, arXiv: 1706.02275. [Online]. Available: <http://arxiv.org/abs/1706.02275>
- [18] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *arXiv:1602.01783 [cs]*, Jun. 2016, arXiv: 1602.01783. [Online]. Available: <http://arxiv.org/abs/1602.01783>
- [19] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing Overestimation Bias in Multi-Agent Domains Using Double Centralized Critics," Dec. 2019, arXiv:1910.01465 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1910.01465>
- [20] C. Yu, A. Velu, E. Vinitksy, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," Nov. 2022, arXiv:2103.01955 [cs]. [Online]. Available: <http://arxiv.org/abs/2103.01955>
- [21] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning," Jun. 2017, arXiv:1706.05296 [cs]. [Online]. Available: <http://arxiv.org/abs/1706.05296>
- [22] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," Jun. 2018, arXiv:1803.11485 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1803.11485>
- [23] S. Nayak, K. Choi, W. Ding, S. Dolan, K. Gopalakrishnan, and H. Balakrishnan, "Scalable Multi-Agent Reinforcement Learning through Intelligent Information Aggregation," May 2023, arXiv:2211.02127 [cs]. [Online]. Available: <http://arxiv.org/abs/2211.02127>
- [24] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine Theory of Mind," Mar. 2018, arXiv:1802.07740 [cs]. [Online]. Available: <http://arxiv.org/abs/1802.07740>
- [25] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling Others using Oneself in Multi-Agent Reinforcement Learning," Mar. 2018, arXiv:1802.09640 [cs]. [Online]. Available: <http://arxiv.org/abs/1802.09640>
- [26] F. Stulp, M. Isik, and M. Beetz, "Implicit coordination in robotic teams using learned prediction models," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. Orlando, FL, USA: IEEE, 2006, pp. 1330–1335. [Online]. Available: <http://ieeexplore.ieee.org/document/1641893/>
- [27] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. A. Ortega, D. J. Strouse, J. Z. Leibo, and N. de Freitas, "Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning," Jun. 2019, arXiv:1810.08647 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1810.08647>
- [28] K. Ndousse, D. Eck, S. Levine, and N. Jaques, "Emergent Social Learning via Multi-agent Reinforcement Learning," Jun. 2021, arXiv:2010.00581 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/2010.00581>
- [29] D. Premack and G. Woodruff, "Does the chimpanzee have a theory of mind?" *Behavioral and Brain Sciences*, vol. 1, no. 4, pp. 515–526, Dec. 1978. [Online]. Available: [https://www.cambridge.org/core/product/identifier/S0140525X00076512/type/journal\\_article](https://www.cambridge.org/core/product/identifier/S0140525X00076512/type/journal_article)
- [30] R. M. Raafat, N. Chater, and C. Frith, "Herdin in humans," *Trends in Cognitive Sciences*, vol. 13, no. 10, pp. 420–428, Oct. 2009. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1364661309001703>
- [31] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement Learning with Unsupervised Auxiliary Tasks," Nov. 2016, arXiv:1611.05397 [cs]. [Online]. Available: <http://arxiv.org/abs/1611.05397>
- [32] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt, "Multi-task Deep Reinforcement Learning with PopArt," Sep. 2018, arXiv:1809.04474 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1809.04474>
- [33] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own Reward: Self-Supervision for Reinforcement Learning," Mar. 2017, arXiv:1612.07307 [cs]. [Online]. Available: <http://arxiv.org/abs/1612.07307>
- [34] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to Navigate in Complex Environments," Jan. 2017, arXiv:1611.03673 [cs]. [Online]. Available: <http://arxiv.org/abs/1611.03673>
- [35] A. Baisero and C. Amato, "Learning Complementary Representations of the Past using Auxiliary Tasks in Partially Observable Reinforcement Learning," *New Zealand*, 2020.
- [36] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," May 2017, arXiv:1705.05363 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1705.05363>
- [37] R. Rădulescu, M. Legrand, K. Efthymiadis, D. M. Roijers, and A. Nowé, "Deep Multi-agent Reinforcement Learning in a Homogeneous Open Population," in *Artificial Intelligence*, M. Atzmueller and W. Duivesteijn, Eds. Cham: Springer International Publishing, 2019, vol. 1021, pp. 90–105, series Title: Communications in Computer and Information Science. [Online]. Available: [https://link.springer.com/10.1007/978-3-03-31978-6\\_8](https://link.springer.com/10.1007/978-3-03-31978-6_8)
- [38] C. Pincioli and *et al.*, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.
- [39] C. Pincioli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2016, pp. 3794–3800. [Online]. Available: <http://ieeexplore.ieee.org/document/7759558/>
- [40] M. Bonani, V. Longchamp, S. Magenat, P. Rétornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, and F. Mondada, "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei: IEEE, Oct. 2010, pp. 4187–4193. [Online]. Available: <http://ieeexplore.ieee.org/document/5649153/>
- [41] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. Montreal, Quebec, Canada: ACM Press, 2009, pp. 1–8. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1553374.1553380>
- [42] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Agent Modeling as Auxiliary Task for Deep Reinforcement Learning," Jul. 2019, arXiv:1907.09597 [cs]. [Online]. Available: <http://arxiv.org/abs/1907.09597>