Heterogeneous Multi-Robot Reinforcement Learning

Matteo Bettini
University of Cambridge
Cambridge, United Kingdom
mb2389@cl.cam.ac.uk

Ajay Shankar
University of Cambridge
Cambridge, United Kingdom
as3233@cl.cam.ac.uk

Amanda Prorok
University of Cambridge
Cambridge, United Kingdom
asp45@cl.cam.ac.uk

ABSTRACT

Cooperative multi-robot tasks can benefit from heterogeneity in the robots' physical and behavioral traits. In spite of this, traditional Multi-Agent Reinforcement Learning (MARL) frameworks lack the ability to explicitly accommodate policy heterogeneity, and typically constrain agents to share neural network parameters. This enforced homogeneity limits application in cases where the tasks benefit from heterogeneous behaviors. In this paper, we crystallize the role of heterogeneity in MARL policies. Towards this end, we introduce Heterogeneous Graph Neural Network Proximal Policy Optimization (HetGPPO), a paradigm for training heterogeneous MARL policies that leverages a Graph Neural Network for differentiable inter-agent communication. HetGPPO allows communicating agents to learn heterogeneous behaviors while enabling fully decentralized training in partially observable environments. We complement this with a taxonomical overview that exposes more heterogeneity classes than previously identified. To motivate the need for our model, we present a characterization of techniques that homogeneous models can leverage to emulate heterogeneous behavior, and show how this "apparent heterogeneity" is brittle in real-world conditions. Through simulations and real-world experiments, we show that: (i) when homogeneous methods fail due to strong heterogeneous requirements, HetGPPO succeeds, and, (ii) when homogeneous methods are able to learn apparently heterogeneous behaviors, HetGPPO achieves higher resilience to both training and deployment noise.

KEYWORDS

Heterogeneity, Multi-agent reinforcement learning, Multi-robot systems

1 INTRODUCTION

Multi-robot systems deployed to tackle complex cooperative tasks can often benefit from heterogeneous physical and/or behavioral traits to fulfill their mission. Such heterogeneous systems have been leveraged in applications such as disaster response [37], collaborative mapping [8], agriculture [26], and package transport [22]. However, synthesizing optimal decentralized policies for these tasks can be computationally hard, and typically scales exponentially with the number of agents [5]. While faster and scalable solutions exist, such as metaheuristics [9], they lack in optimality. Multi-Agent Reinforcement Learning (MARL) [61] can be used as a scalable approach to find near-optimal solutions to these problems. However, MARL algorithms without inter-agent communication cannot be easily applied to real-world robotic problems, where partial observability of individual agents is pervasive. Communication is key to overcoming this partial observability, and to enable cooperation.

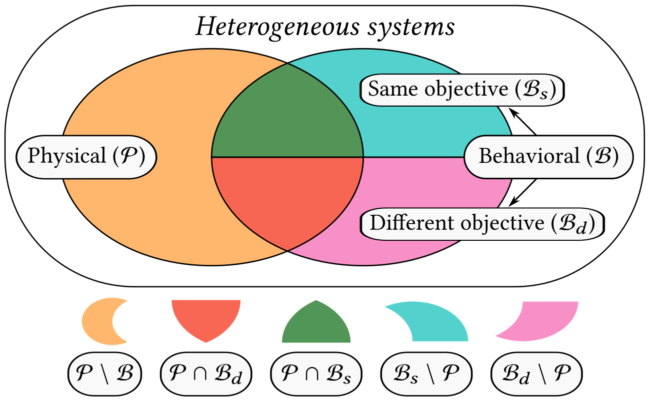

Heterogeneous systems
Physical ($\mathcal{P}$)
Same objective ($\mathcal{B}_s$)
Behavioral ($\mathcal{B}$)
Different objective ($\mathcal{B}_d$)
$\mathcal{P} \setminus \mathcal{B}$
$\mathcal{P} \cap \mathcal{B}_d$
$\mathcal{P} \cap \mathcal{B}_s$
$\mathcal{B}_s \setminus \mathcal{P}$
$\mathcal{B}_d \setminus \mathcal{P}$

Figure 1: Taxonomy of heterogeneous multi-robot/agent systems. *Top*: the three heterogeneity classes ($\mathcal{P}$, $\mathcal{B}_s$, $\mathcal{B}_d$). *Bottom*: the five mutually exclusive heterogeneity subclasses. Every heterogeneous system belongs to one of these subclasses.

Our work deals with *heterogeneous multi-robot reinforcement learning*, a paradigm located at the boundary of MARL (with inter-agent communication) and heterogeneous multi-robot systems.

Most cooperative MARL works constrain agents to share policy neural network parameters to improve training sample efficiency [24, 44, 53]. This causes the agents' models to be identical and, thus, homogeneous. While this is beneficial to speed-up training, it can prevent learning in scenarios that require heterogeneous behavior. A classical method of overcoming this imposed homogeneity is to include a unique integer (e.g., the agent's index) as part of each agent's observations [19, 24]. This allows the agents to share the same policy while exhibiting *apparently* different behavior. Despite its wide adoption, this solution has many drawbacks [12].

We are interested in learning *truly* heterogeneous decentralized MARL policies. While it is common practice to learn heterogeneous policies when optimizing for different objectives [33], there is a dearth of work in applying this paradigm to scenarios where the objective is shared. Current solutions are few and tailored to specific tasks, and, as such, do not address the broader study and categorization of heterogeneity in MARL. Furthermore, they are limited to noise-free videogame-like MARL benchmarks [30, 45], without considering real-world multi-robot tasks with inter-agent communication. Therefore, we need a framework that enables *true* heterogeneity among communicating MARL agents and can learn policies that run in a decentralized fashion for (real-world) heterogeneous multi-robot systems.

In this work, we introduce Heterogeneous Graph Neural Network Proximal Policy Optimization (HetGPPO), a paradigm for heterogeneous MARL that overcomes the aforementioned issues. HetGPPO is a framework for training heterogeneous MARL policies that leverages a Graph Neural Network (GNN) for differentiable inter-agent communication. Our architecture enables learning for heterogeneous agents while being conditioned only on local communication and local observations. This enables to train HetGPPO in a decentralized fashion, in-line with the Decentralized Training Decentralized Execution (DTDE) paradigm [25].

We begin by presenting a taxonomy of heterogeneous systems in Sec. 2. The purpose of this taxonomy is to classify such systems according to the source of their heterogeneity. We use this taxonomy in Sec. 3 to categorize related works in the domains of multi-robot systems and MARL. Sec. 4 formulates the MARL problem tackled in this paper. In Sec. 5, we introduce HetGPPO and its homogeneous counterpart GPPO. To motivate the need for policy heterogeneity, we distill and define the techniques that homogeneous models use to emulate heterogeneous behavior (Sec. 6). Through example scenarios, we demonstrate how these technique work and how they can prove brittle in real-world conditions when compared to truly heterogeneous models. Finally, in Sec. 7, we present evaluations of our framework both in simulated and real-world multi-robot cooperative scenarios. These show that: (i) when homogeneous methods fail due to strong heterogeneous requirements, HetGPPO succeeds, and, (ii) when homogeneous methods are able to learn apparently heterogeneous behaviors, HetGPPO achieves higher resilience to both training and deployment noise. Furthermore, our real-robot experiments demonstrate how heterogeneous policies are intrinsically more resilient to real-world conditions.

In this paper, we demonstrate the power of heterogeneous MARL applied to real-world multi-robot systems. We claim the following key contributions:

(1) A taxonomy of heterogeneous systems that jointly categorizes research in the multi-robot and multi-agent domains;
(2) A discourse on *behavioral typing* techniques that homogeneous models rely on to emulate heterogeneous behavior, with empirical evidence for their brittleness in deployment;
(3) HetGPPO, a MARL model able to learn heterogeneous communicating policies in a decentralized fashion; and,
(4) Detailed evaluations of the performance and resilience of heterogeneous policies compared to homogeneous ones in several cooperative multi-robot tasks, both through simulations and real-world experiments.

2 TAXONOMY OF HETEROGENEOUS SYSTEMS

In spite of a substantial body of work attempting to stimulate research on heterogeneous systems (see [1] and the references therein), the robotics and learning community still lacks a shared and structured taxonomy of heterogeneous systems. To properly characterize the related works in the heterogeneity (diversity) domain, we introduce a taxonomy of heterogeneous systems, shown in Fig. 1. According to our taxonomy, system heterogeneity is categorized in two classes: *Physical* ($\mathcal{P}$) and *Behavioral* ($\mathcal{B}$).

A team is considered *physically* ($\mathcal{P}$) heterogeneous when at least one of its components (i.e., agents, robots) differs from the others in terms of hardware or physical constraints. That is, it might have different sensors, actuators, motion constraints, etc. These physical differences might lead to different capabilities. For example, a small drone might be able to fly and move aggressively, but likely has shorter battery life than a big and slow ground robot. This type of heterogeneity can lead to different observation and action spaces in the context of learning, for example when robots are equipped with different sensors or actuators.

A team is considered *behaviorally* ($\mathcal{B}$) heterogeneous when at least one of its components differs from the others in terms of software or behavioral model. That is, two behaviorally heterogeneous agents can produce distinct policy outputs when observing the same input. For example, two physically identical drones might cooperate to monitor a site: here, one drone can survey from far away and direct the other to areas that need closer inspection. Behavioral heterogeneity is divided in two: *Same objective* ($\mathcal{B}_s$) and *Different objective* ($\mathcal{B}_d$). In $\mathcal{B}_s$ heterogeneous systems, agents optimize the same objective function through heterogeneous behavior. In MARL, this means that they share the same (global or local) reward function. $\mathcal{B}_s$ heterogeneous systems usually represent cooperative settings [11]. However, they could also model adversarial scenarios where agents with the same objective compete for limited resources [7]. In $\mathcal{B}_d$ heterogeneous systems, agents optimize different objective functions through heterogeneous behavior. In MARL, this means that they have different local reward functions or a global reward deriving from the composition of such local functions. $\mathcal{B}_d$ heterogeneous systems usually represent non-cooperative or adversarial settings [33]. However, they could also model cooperative scenarios where agents optimize different sub-functions for a higher-order task [12]. For example, in cooperative search and rescue scenarios, one robot might only be tasked to remove debris, while the others are tasked with the search in an uncluttered space.

Physical and behavioral heterogeneity are not mutually exclusive. Thus, the three heterogeneity classes introduced ($\mathcal{P}$, $\mathcal{B}_s$, $\mathcal{B}_d$) delineate five heterogeneity subclasses that a system can belong to:

- $\mathcal{P} \setminus \mathcal{B}$: Agents are physically different but share the same behavioral model.
- $\mathcal{P} \cap \mathcal{B}_d$: Agents are physically different and differ in behavioral models and objectives.
- $\mathcal{P} \cap \mathcal{B}_s$: Agents are physically different and differ in behavioral models, but share the same objective.
- $\mathcal{B}_s \setminus \mathcal{P}$: Agents are physically identical and share the same objective but differ in behavioral models.
- $\mathcal{B}_d \setminus \mathcal{P}$: Agents are physically identical but differ in behavioral models and objectives.

While this taxonomy is concerned with classifying heterogeneous systems, it does not attempt to measure the *degree* of heterogeneity. Furthermore, it represents a *high-level* classification and does not consider dynamic $\mathcal{P}$ heterogeneity, such as different battery levels or hardware deterioration [50].

3 RELATED WORK

In this section, we review the current state of the art in the area of heterogeneous multi-robot/agent systems. We classify the related works according to our taxonomy in Tab. 1.

3.1 Heterogeneity in multi-robot systems

The core literature on heterogeneous robotics has generally focused on developing coordination algorithms that leverage the physical heterogeneity of a team to their advantage. Therefore,

Table 1: Related work in heterogeneous multi-robot/agent systems classified according to our taxonomy of Sec. 2.

Heterogeneity class	Multi-robot systems	MARL
$\mathcal{P} \setminus \mathcal{B}$	[8]	[55],[54]
$\mathcal{P} \cap \mathcal{B}_d$	[37]	[33],[12]
$\mathcal{P} \cap \mathcal{B}_s$	[17],[40],[18],[36],[39], [43],[42],[46],[28],[34], [35],[10],[14]	[49]
$\mathcal{B}_s \setminus \mathcal{P}$	[1],[4],[2],[3],[31], [52],[56]	[57],[11],[58]
$\mathcal{B}_d \setminus \mathcal{P}$	[23],[47]	[33],[12]

these works fall in the $\mathcal{P} \cap \mathcal{B}_s$ class. Such diversity can manifest itself in the form of different sensor ranges [42], diverse sensing capabilities [46], or different maximum speeds [28]. These differences can then be exploited in a variety of problems such as multi-robot coverage [28, 42, 46] and heterogeneous task assignment [40, 43], with resilient formulations that can handle uncertainties in robot capabilities [17] or the environment [18, 36, 39]. Sensor heterogeneity has also received attention in the context of active sampling and mapping [34, 35], where heterogeneous computational resources can impact task execution [10]. Lastly, $\mathcal{P} \cap \mathcal{B}_s$ diversity has also been investigated in more complex problems such as heterogeneous trajectory planning [14].

Interestingly, such physical diversity without behavioral diversity ($\mathcal{P} \setminus \mathcal{B}$) can often represent a *constraint* for the problem. Works in this heterogeneity class try to behaviorally reconcile the physical heterogeneity of robots in order to apply homogeneous solutions to the problem at hand. Heterogeneous multi-robot SLAM is an example application where scans coming from different robots, equipped with diverse sensors, need to be matched in order to build a homogeneous shared map [8].

Behavioral heterogeneity for physically identical robots is a less explored but promising research direction [1]. Works in this area mostly tackle cooperative problems, leveraging $\mathcal{B}_s \setminus \mathcal{P}$ heterogeneity. Early research by Balch [2, 3] and Li et al. [31] focuses on learning behavioral specialization for multi-robot teams using RL. Game-theoretic autonomous racing [52, 56] constitutes an adversarial setting of $\mathcal{B}_s \setminus \mathcal{P}$ heterogeneity. Note that game-theoretic controllers do not present heterogeneous behavior when all players use the symmetric Nash equilibrium strategy [38]. However, heterogeneity emerges when some robots in the team use traditional model predictive controllers.

Conversely, heterogeneous behavior with different objectives ($\mathcal{B}_d$) has also been analyzed for cooperative robotic tasks, for instance, by dividing a global task into sub-tasks for groups of identical robots ($\mathcal{B}_d \setminus \mathcal{P}$) [23, 47]. When the robots additionally have physical differences between sub-groups, these differences can be leveraged to tackle complex multi-robot tasks, such as post-disaster collaborative mapping [37], resulting in $\mathcal{P} \cap \mathcal{B}_d$ heterogeneity.

All the works discussed in this subsection focus on a given heterogeneity class and problem, and develop a targeted solution for that setting. To a large extent, the approaches leverage conventional control theoretical methods. Our work, in contrast, proposes a learning-based framework to synthesize communicating multi-agent/robot policies, and can be applied to any heterogeneity class.

3.2 Heterogeneity in MARL

MARL has recently gained increasing traction as an effective technique to tackle multi-robot problems [61]. Using MARL, it is possible to synthesize efficient decentralized multi-agent controllers for hard coordination problems [5]. Homogeneous policies (that share parameters) for physically identical agents are abundant in MARL [20, 24, 29, 44, 53] and constitute the core of the research literature. In an attempt to emulate heterogeneous behavior, a common practice is to augment each agent's observation space with a unique index that represents the agent's type [19, 24]. In this case, agents share the same homogeneous multimodal policy, conditioned on a unique constant index. We define and discuss in depth the limitations of this approach in Sec. 6. $\mathcal{P} \setminus \mathcal{B}$ heterogeneity in MARL focuses on leveraging the power of parameter sharing and homogeneous training for physically different agents. This is achieved by mapping heterogeneous observation spaces into homogeneous fixed-length encodings [55], or by padding and including the agent index into observations [54].

The majority of heterogeneous MARL literature falls in the $\mathcal{B}$ heterogeneity class. Different behavioral roles for physically identical agents can be learned through various techniques, such as conditioning agents' policies on a latent representation [57], decomposing and clustering action spaces [58], or by an intrinsic reward that maximizes the mutual information between the agent's trajectory and its role [11]. All the aforementioned works consider physically identical agents with the same objective, thus leveraging $\mathcal{B}_s \setminus \mathcal{P}$ heterogeneity. Furthermore, they do not use inter-agent communication, and hence their application to highly partially observable coordination problems is limited. When considering physically different robots, heterogeneous action or observation spaces have to be taken into account. Such $\mathcal{P} \cap \mathcal{B}_s$ heterogeneity with communicating agents can be modeled, for instance, by an ad-hoc GNN layer for each physically different robot type [49]. While this may be suitable for some tasks where robot types are known beforehand, it prevents physically identical agents from learning heterogeneous behavior.

Behavioral heterogeneity with different objectives ($\mathcal{B}_d$) emerges due to different agent reward functions, as discussed in Sec. 2. MADDPG [33] uses this paradigm in a centralized training approach to learn individual (not shared) actors and critics. They test their approach in mixed cooperative-competitive tasks. In these tasks, both physically identical and physically different agents (i.e., different maximum speeds) are considered. Thus, MADDPG leverages heterogeneity classes $\mathcal{B}_d \setminus \mathcal{P}$ and $\mathcal{P} \cap \mathcal{B}_d$. The same heterogeneity classes are studied in [12], which proposes to use parameter sharing among sub-groups of agents which are physically identical and share the same reward function. This approach, however, prevents physically identical agents with the same objective to employ different behavioral roles to solve a task.

Most works discussed in this section propose solutions to problems that sit exclusively within one given heterogeneity subclass. While a selected few could be applied to multiple classes [11, 33, 57], they leverage centralized training methods and do not consider inter-agent communication. These are two key features needed to make MARL suitable for multi-robot problems.

4 PROBLEM FORMULATION

We now formulate the multi-robot MARL problem tackled in this work. To do so, we first introduce the multi-agent extension of a Partially Observable Markov Decision Process (POMDP) [27].

Partially Observable Markov Games. A Partially Observable Markov Game (POMG) is defined as a tuple

$$\left\langle \mathcal{V}, \mathcal{S}, O, \{\sigma_i\}_{i\in\mathcal{V}}, \mathcal{A}, \{\mathcal{R}_i\}_{i\in\mathcal{V}}, \mathcal{T}, \gamma \right\rangle,$$

where $\mathcal{V} = \{1, \dots, n\}$ denotes the set of agents, $\mathcal{S}$ is the state space, shared by all agents, and, $O \equiv O_1 \times \ldots \times O_n$ and $\mathcal{A} \equiv \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ are the observation and action spaces, with $O_i \subseteq \mathcal{S},\ \forall i \in \mathcal{V}$. Further, $\{\sigma_i\}_{i\in\mathcal{V}}$ and $\{\mathcal{R}_i\}_{i\in\mathcal{V}}$ are the agent observation and reward functions[1], such that $\sigma_i : \mathcal{S} \mapsto O_i$, and, $\mathcal{R}_i : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \mapsto \mathbb{R}$. $\mathcal{T}$ is the stochastic state transition model, defined as $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$. Lastly, γ is the discount factor.

We structure the agents in a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Nodes $i \in \mathcal{V}$ represent agents and edges $e_{ij} \in \mathcal{E}$ represent communication links. The set of edges is dependent of the maximum agent communication range and changes over time. The communication neighborhood of each agent is defined as $\mathcal{N}_i \equiv \{v_j \mid e_{ij} \in \mathcal{E}\}$.

At each timestep t, each agent i gets an observation $o_i^t = \sigma_i(s^t) \in O_i$ that is a portion of the global state $s^t \in \mathcal{S}$. This is communicated to the neighboring agents $\mathcal{N}_i^t$. A stochastic policy π_i uses this information to compute an action $a_i^t \sim \pi_i(\cdot|o_{\mathcal{N}_i}^t)$. The agents' actions $\mathbf{a}^t = (a_1^t, \dots, a_n^t) \in \mathcal{A}$, along with the current state s^t, are then used in the transition model to obtain the next state $s^{t+1} \sim \mathcal{T}\left(\cdot|s^t, \mathbf{a}^t\right)$. A reward $r_i^t = \mathcal{R}_i\left(s^t, \mathbf{a}^t, s^{t+1}\right)$ is then fed to agent i.

The goal of each agent is to maximize the sum of discounted rewards $v_i^t = \sum_{k=0}^{T} \gamma^k r_i^{t+k}$ over an episode with horizon T, potentially infinite[2]. v_i^t is called the return. Each agent has a value function $V_i(o_{\mathcal{N}_i}) = \mathbb{E}_{\pi_i}\left[v_i^t \middle| o_{\mathcal{N}_i}^t = o_{\mathcal{N}_i}\right]$, which represents the expected return starting from observations $o_{\mathcal{N}_i}$ and following policy π_i. This function estimates the "goodness" of an observation. In this work, we use the Proximal Policy Optimization (PPO) actor-critic algorithm [48], which approximates the policy (actor) and the value function (critic) using neural networks and a constrained policy gradient update.

Problem. Learn heterogeneous policies $\pi_i(o_{\mathcal{N}_i}^t; \theta_i)$ and critics $V_i(o_{\mathcal{N}_i}; \theta_i)$ conditioned on the neural network parameters θ_i, different for each agent. The observations $o_{\mathcal{N}_i}^t$ from the agent's neighborhood $\mathcal{N}_i$ are obtained through a differentiable communication channel, making learning inherently decentralizable.

Our objective is to crystallize the role of heterogeneity in MARL policies. Towards this end, we develop a model that addresses the problem description above, motivating it with an empirically-backed discourse on the shortcomings of homogeneous policies (and the *behavioral typing* techniques that they rely on).

[1]Note that, while we formulate our problem with local agent reward functions $\mathcal{R}_i$ (to enable learning of $\mathcal{B}_d$ heterogeneity), our experiments all present a global reward function $\mathcal{R} = \mathcal{R}_1 = \ldots = \mathcal{R}_n$ which cannot be decomposed into local sub-functions. This reward encodes a global cooperative objective, leading to $\mathcal{B}_s$ heterogeneity.

[2]γ^k indicates γ to the power of k, and not the timestep superscript.

5 HETEROGENEOUS MODEL

We introduce the two MARL models that constitute the methodology leveraged in this work: Graph Neural Network Proximal Policy Optimization (GPPO) and its heterogeneous counterpart, HetGPPO.

GPPO builds upon Independent Proximal Policy Optimization (IPPO) [13]. In IPPO, each agent learns a local critic $V_i(o_i)$ and actor $\pi_i(o_i)$, conditioned only on its own observations. Conditioning the critic only on local observations and not on the full state s introduces non-stationarity during training. This results in other agents being considered as part of the environment and not explicitly modeled in the critic. While this can be problematic, it has the advantage of not requiring global information during training. Furthermore, IPPO has been shown to outperform many fully-observable critic models on state-of-the-art MARL benchmarks [13].

GPPO overcomes the limitations of IPPO while maintaining its benefits. It uses a GNN communication layer, allowing agents to share information in neighborhoods to coordinate and overcome partial observability. Thanks to this, the GPPO critic $V_i(o_{\mathcal{N}_i})$ and actor $\pi_i(o_{\mathcal{N}_i})$ are conditioned on local communication neighborhood observations $o_{\mathcal{N}_i}$. This helps overcome non-stationarity, while requiring only local information and communication during training.

The GPPO model is illustrated in Fig. 2. At each timestep, each agent i observes the environment, collecting the observations o_i. These observations contain absolute geometrical features, such as the agent position $\mathbf{p}_i \in \mathbb{R}^2$. The non-absolute features are passed through a Multi Layer Perceptron (MLP) encoder, obtaining an embedding z_i. The absolute position and the agent velocity $\mathbf{v}_i \in \mathbb{R}^2$ are used to compute edge features e_{ij}, which are relative features of agents i and j. In this work, we use the relative position $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ and relative velocity $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ as edge features $e_{ij} = \mathbf{p}_{ij} \mathbin{||} \mathbf{v}_{ij}$, where $||$ indicates the concatenation operation. This process ensures that GNN outputs are invariant to translations in $\mathbb{R}^2$ (i.e., the same output is obtained if all the team is translated in space), helping the model generalize [21]. The edge features e_{ij} and the agent embedding z_i are then used in the message-passing GNN kernel:

$$h_i = \psi_{\theta_i}(z_i) + \bigoplus_{j\in\mathcal{N}_i} \phi_{\theta_i}(z_j \mathbin{||} e_{ij}).$$

Here, ψ_{θ_i} and ϕ_{θ_i} are two MLPs, parameterized by the agent parameters θ_i[3], and $\bigoplus$ is an aggregation operator (e.g., sum). The GNN output h_i is then fed to two different MLP decoders, which output the action $a_i \sim \pi_i(\cdot|o_{\mathcal{N}_i})$ and the value $V_i(o_{\mathcal{N}_i})$. Similar to IPPO, GPPO uses parameter sharing to improve sample efficiency. Thus $\theta_1 = \ldots = \theta_n$. Parameter sharing allows agents to benefit from collective experiences and thereby reduces training time. On the other hand, it enforces centralized training and constraints agents' policies to be identical (i.e., homogeneous).

HetGPPO, *removes* the parameter sharing constraint of GPPO, thus allowing agent policies to diverge, $\theta_1 \neq \ldots \neq \theta_n$. However, the impact of not sharing parameters in the context of GNN communications is profound: the permutation equivariance property of GNNs [59] does not hold, since the agents now learn different message encoding and interpreting strategies. This results in the GNN having to learn a different team output for all the possible permutations of a given team input, instead of learning only one

[3]With θ_i we indicate the parameters for all the neural network layers of agent i.

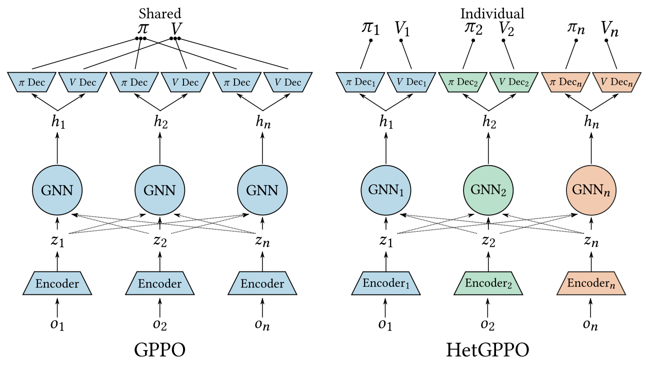


Figure 2: Architecture of GPPO and HetGPPO: MARL models with communicating agents. Each agent passes its observation through an encoder, then aggregates messages received from its neighbors using a translation-invariant message-passing GNN and updates its hidden state h_i. h_i is then used as input to the policy and value decoders (Dec). HetGPPO is equivalent to GPPO without parameter sharing.

output. This can lead to decreases in generalization power and sample efficiency. On the other hand, gradients are backpropagated through communication neighborhoods, enabling agents to learn collectively from local interactions.

The structure of HetGPPO, shown in Fig. 2, allows for *Decentralized Training Decentralized Execution (DTDE)*. This is thanks to the fact that GPPO critics are not conditioned on global information. While GPPO uses parameter sharing, HetGPPO removes this need, thus enabling training in any environment where just inter-agent communication is possible. We note that, by implementing an ad-hoc mechanism to achieve decentralized parameter sharing (e.g., through distributed optimization [60]), GPPO could be trained in a decentralized fashion as well.

We implement HetGPPO and GPPO in PyTorch [41] and employ the RLlib [32] framework for training. The code is available here[4]. Simulations are executed in custom created scenarios using the VMAS simulator [6], available at this link[5].

6 BEHAVIORAL TYPING

HetGPPO, introduced above, allows us to learn *truly* heterogeneous policies. Counter-intuitively, it is also possible to learn *apparently* heterogeneous behavior with homogeneous models like GPPO. This allows agents to emulate heterogeneous behavior while leveraging the sample efficiency benefits of parameter sharing. A shared model can encompass different behavioral types which are activated by particular combinations of the input observations. For example, if two robots are transporting a package towards a destination, the model can identify if an agent is in the back (further from the goal) and assign it a different behavioral type from that of the agent in the front. The input observations provide the conditions for the model to assign behavioral types to the agents.

We refer to this identification process as *typing*. Fig. 3 depicts a classification of behavioral typing techniques which we describe

[4]https://github.com/proroklab/HetGPPO

[5]https://github.com/proroklab/VectorizedMultiAgentSimulator

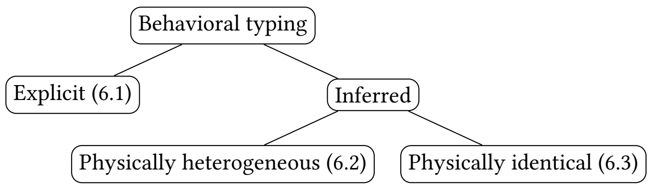


Figure 3: Different forms of behavioral typing. Homogeneous policies use typing to differentiate among agents and emulate heterogeneous behavior.

in the following subsections. Note that behavioral types lie in a continuous behavioral space (and are not part of a discrete set) [2].

6.1 Explicit behavioral typing

The most popular form of behavioral typing consists in feeding the index i of the agent explicitly as part of the observation. This practice has been used extensively in the MARL literature [12, 19, 24, 54]. However, it requires the model to learn a multimodal policy, which switches modes based on this integer index. This can lead to discontinuities in the agents' policy and has been shown to perform sub-optimally [12].

Definition 6.1 (Explicit behavioral typing). Explicit behavioral typing occurs when a shared decentralized MARL policy is able to type agents based on a constant value concatenated to the input, different for each agent.

When no explicit index is available, a shared policy may still be able to emulate heterogeneous behavior [15]. We refer to this phenomenon as *inferred* behavioral typing. Inferred typing can occur for both physically heterogeneous and physically identical agents.

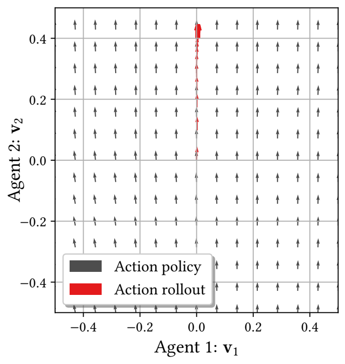
6.2 Inferred behavioral typing for physically heterogeneous agents

We first present a case study of inferred behavioral typing for agents that are physically heterogeneous.

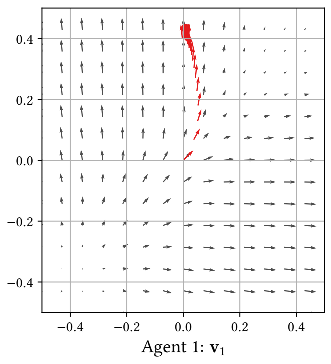
Definition 6.2 (Inferred behavioral typing for physically heterogeneous agents). Inferred behavioral typing for physically heterogeneous agents occurs when a shared decentralized MARL policy is able to type $\mathcal{P}$ heterogeneous agents through their observations.

Scenario A (Fig. 4). Consider two robots with different masses, $m_1 > m_2$, located in a 1D workspace at random positions. The robots observe their own position $\mathbf{p}_i \in \mathbb{R}$ and velocity $\mathbf{v}_i \in \mathbb{R}$ and share them via communication. Their action is a force $\mathbf{f}_i \in \mathbb{R}$. They are rewarded collectively to maximize the maximum speed in the team while minimizing the energy consumed. The optimal policy in this case is, clearly, for the robot with the higher mass to not move at all, while the lighter robot moves at the maximum speed. Evidently these behaviors are heterogeneous, since $\mathbf{f}_1 \neq \mathbf{f}_2$ when both agents receive the same observations.

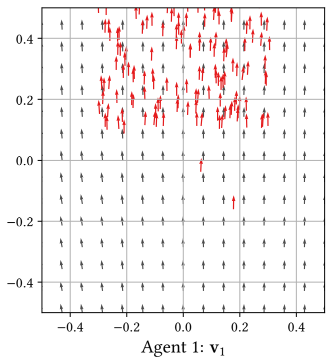
We train the agents in this scenario using GPPO and HetGPPO. Fig. 4 shows a graphical representation of the learned policies of each model. In these plots, an arrow represents the team action vector $\vec{\mathbf{f}}(\vec{\mathbf{v}}) = [\mathbf{f}_1(\vec{\mathbf{v}}), \mathbf{f}_2(\vec{\mathbf{v}})]$ as a function of the observation $\vec{\mathbf{v}} =$



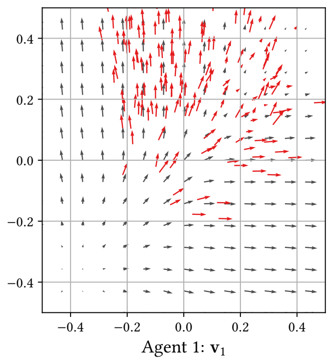
(a) Heterogeneous



(b) Homogeneous



(c) Heterogeneous with noise



(d) Homogenous with noise

Figure 4: Policies learned for Scenario A represented as vector fields (gray) and rollouts in the environment (red). (a) and (b) are not subject to deployment noise. (c) and (d) are subject to ± 0.3 uniform noise on the observations. In these plots, an arrow represents the team action vector $\vec{f}(\vec{v}) = [f_1(\vec{v}), f_2(\vec{v})]$ as a function of the observation $\vec{v} = [v_1, v_2]$. The rollouts always start in the origin ($\vec{v} = [0, 0]$). We can observe how the vector field representing the homogeneous policies is forced to be invariant to permutations of the two inputs and thus is symmetric along $v_1 = v_2$. This causes it to become brittle in the presence of noise (d), which makes the observations fall in the wrong part of the plane where the symmetry enforces a suboptimal policy (horizontal arrows).

$[\mathbf{v}_1, \mathbf{v}_2]$. A vertical arrow at $[0, 0]$ indicates that, when the agents are both still, agent 1 wants to stay still while agent 2 wants to increase its velocity. We plot the policy mean action[6] for every observation pair with a gray vector field, and show a rollout of the policy in red. In Fig. 4a, we observe how HetGPPO is able to learn the optimal policy, which is not dependent on any observation. Thanks to physically inferred typing, GPPO (Fig. 4b) is surprisingly also able to learn a policy that grants optimal rollouts. We observe how, due to homogeneity, the GPPO policy is forced to be symmetric about the $\mathbf{v}_1 = \mathbf{v}_2$ axis. Thus, when the GPPO agents are spawned at $[0, 0]$, they forcibly take the same action of increasing their velocities. Due to their $\mathcal{P}$ differences, however, this action produces different velocities, making the rollout quickly diverge from the symmetry and thus producing optimal behavior. The fact that a physical difference (i.e., different agent mass) produces different observations (i.e., different agent speed) for the same action, enables the homogeneous model to learn an optimal policy with apparent heterogeneous behavior – an example of *inferred behavioral typing for physically heterogeneous agents*.

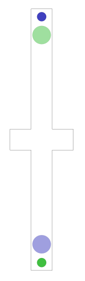
Physically inferred typing proves to be a brittle solution. When additive uniform observation noise is injected during rollouts, we observe how the HetGPPO rollout (Fig. 4c) is not impacted at all, while the GPPO rollout (Fig. 4d) occasionally falls on the other side of the diagonal, producing the symmetrical opposite of the optimal behavior, and causing the heavy agent to move (horizontal arrows).

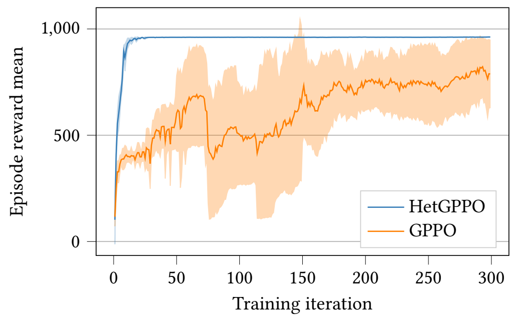
6.3 Inferred behavioral typing for physically identical agents

We now present a case study of inferred behavioral typing for agents that are physically identical.

Definition 6.3 (Inferred behavioral typing for physically identical agents). Inferred behavioral typing for physically identical agents occurs when a shared decentralized MARL policy is able to type physically identical agents through their observations.

[6]The PPO policy is stochastic and outputs a distribution over $\mathbf{f}_i$.


(a) Scenario

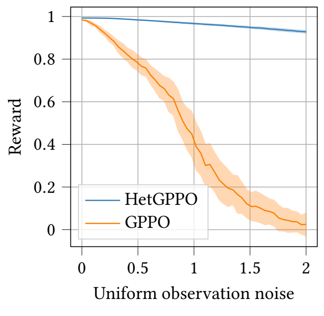


(b) Training performance

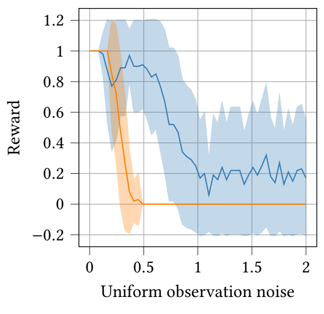
Figure 5: Scenario B. (a): The setup with two robots (bigger circles) on opposite sides of a corridor which need to give way to each other to reach their goals (smaller circles). (b) The training curve for Scenario B, showing that, while the heterogeneous model is able to solve the scenario immediately, homogeneous models need around 300 training iterations to learn inferred behavioral typing for physically identical agents. We plot the mean and standard deviation of 10 different runs. Each iteration is performed over 200 episodes.

Scenario B (Fig. 5). Consider now two physically identical robots, initialized at different ends of a narrow corridor, depicted in Fig. 5a. Each robot is positioned in front of the other's goal. The corridor is wide enough to fit only one robot, but contains two robot-sized recesses in the center. The robots observe and communicate their respective 2D positions and velocities, and are tasked with reaching their goals without colliding. Thus, the task can only be solved when one robot gives way to the other.

Again, we train the agents with both GPPO and HetGPPO in this scenario. By looking at the training reward plot in Fig. 5b,

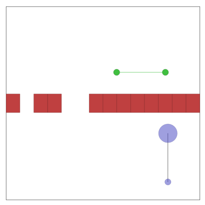


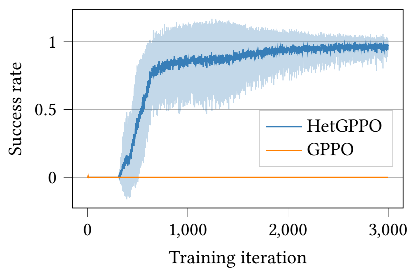
(a) Scenario A



(b) Scenario B

Figure 6: Performance of homogeneous and heterogeneous models in the presence of deployment noise on the two inferred typing scenarios. Reward is normalized between 0 and 1. Uniform noise is applied to all observations and it is in the same units as the observations. We report the mean and standard deviation of the normalized reward on 100 runs for 50 noise values between 0 and 2.


(a) Scenario



(b) Training performance

Figure 7: Performance evaluation in the passage scenario with differently sized robots. Here, the homogeneous model is not able to perform *inferred behavioral typing for physically heterogeneous agents* since $\mathcal{P}$ heterogeneity does not affect the robots' observations. Thus, only the heterogeneous model is able to solve the task. We plot the mean and standard deviation success rate of 4 runs. Each iteration is performed over 200 episodes of experience.

we can see that both models are able to learn the correct behavior (reward > 700). GPPO leverages *inferred typing for physically identical agents* and is able to assign the "give way" role dynamically according to the relative position and velocity of the two robots. However, we observe that learning behavioral typing takes all 300 training iterations, while the heterogeneous model learns the optimal solution with only 20 iterations.

6.4 Limitations of behavioral typing

Although homogeneous models can use behavioral typing to learn apparently heterogeneous behavior, this does not prove to be a reliable and scalable solution.

In [12] it is shown that the performance of explicit behavioral typing degrades as a function of the number of types to be learned. Furthermore, the authors empirically show that this performance decrease is not related to the capacity of the shared homogeneous model (i.e., the number of parameters).

Inferred indexing also proves to be a brittle solution. To characterize this brittleness, we perform an evaluation by injecting observation noise during execution. This is shown in Fig. 6. We report the mean and standard deviation of the normalized reward on 100 runs for 50 noise values between 0 and 2. As we can observe, all models start with the optimal policy with a reward of 1 when the noise is 0. As the noise increases, we observe how homogeneous models either almost immediately lose functionality (like for Scenario B in Fig. 6b), or degrade in performance rapidly (like for Scenario A in Fig. 6a). A heterogeneous policy, in contrast, is able to tolerate higher magnitudes of noise, and, even in the difficult corridor scenario, still manages to complete the task about 20% of the time at high noise values.

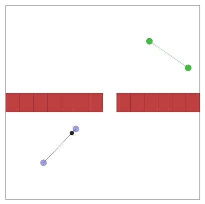
7 EXPERIMENTAL EVALUATIONS

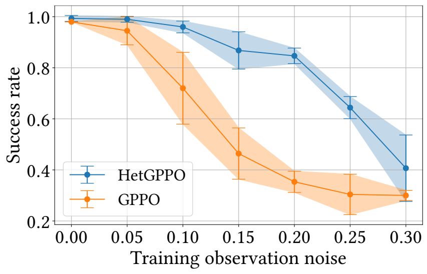
We now present some evaluations of the proposed models in simulated and real environments.

Performance evaluation. We evaluate HetGPPO on a simulated 2D task which requires heterogeneous behavior. The task is shown in Fig. 7a. Here, two robots of different sizes (blue circles), connected by a rigid linkage through two revolute joints, need to cross a passage while keeping the linkage parallel to it and then match the desired goal position (green circles) on the other side. The passage is comprised of a bigger and a smaller gap, which are spawned in a random position and order on the wall, but always at the same distance between each other. The team is spawned in a random order and position on the lower side with the linkage always perpendicular to the passage. The goal is spawned horizontally in a random position on the upper side. Each robot observes and communicates its velocity, relative position to each gap, and relative position to the goal center. The shaped global reward is composed of two convex terms. Before the passage, the robots are rewarded to keep the linkage parallel to the goal and to bring its center to the center of the passage. After the passage, the robots are rewarded for bringing it to the goal at the desired orientation. Collisions are also penalized.

Fig. 7b shows training success rate (i.e., percentage of episodes in each batch that complete the task). The heterogeneous model is able to learn two behaviorally different policies: the bigger robot passes through the bigger gap and the smaller robot through the smaller gap, achieving the optimal solution. On the other hand, the homogeneous model is not able to assign these two behavioral types using *inferred behavioral typing for physically heterogeneous agents*, since the $\mathcal{P}$ heterogeneity caused by different robot sizes does not affect the robots' observations. Agents with homogeneous policies never manage to cross the passage, being deterred by unavoidable collisions.

Resilience to training noise. As elucidated in Sec. 6, homogeneous models can learn heterogeneous behavior. In this subsection, we evaluate the resilience of this paradigm in the presence of observation noise during training. We consider the task depicted in Fig. 8a. This is the same as in Fig. 7a with the difference that the robots are now physically identical, but the linkage has an asymmetric mass (black circle) that causes a different type of $\mathcal{P}$ heterogeneity, reflected in the velocity observations. The passage is a single gap, positioned randomly on the wall. The agents need to cross it while

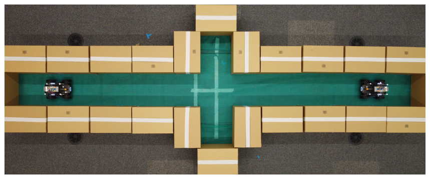

(a) Scenario

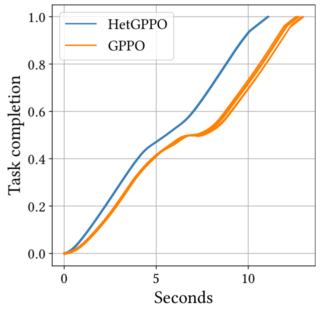


(b) Resilience to training noise

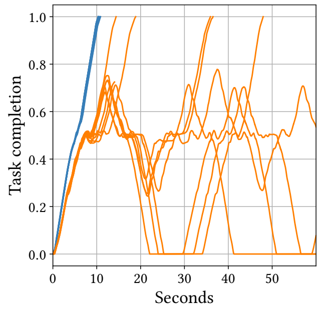
Figure 8: Resilience to uniform observation noise during training in the passage scenario with asymmetric package. Here, the heterogeneous model is able to maintain higher performance as the noise increases. We train the two models for 7 different noise values. For each noise value, we report the mean and standard deviation of the success rate after 1000 training iterations for 5 runs. Each training iteration is performed over 200 episodes of experience.

(a) Robot


(b) Scenario B (real world)



(c) Simulation



(d) Real world

Figure 9: Real-world deployment of Scenario B (Fig. 5). We report 10 runs for each model both in simulation and in the real world. We plot task completion (the scaled sum of the negative distances of each robot from its goal) over time. While in simulation both models are able to perform the task, real world imperfections make the homogeneous model dynamically switch between learned behavioral types, leading to the robots switching positions multiple times near the central area. This causes the zigzag behavior in (d) with certain rollouts failing or taking over the maximum allocated time of 60s.

keeping the linkage perpendicular to the wall and avoiding collisions. The team and the goal are spawned in a random position, order, and rotation on opposite sides of the passage.

In Fig. 8b we report the training success rate for different observation noise values. Thanks to *inferred behavioral typing for physically heterogeneous agents* we see that both models solve the task optimally when 0 noise is added. As noise increases, the heterogeneous model is able to maintain significantly better performance. For example, with 0.2 observation noise, HetGPPO still achieves more than 80% success rate, while GPPO is below 40%.

Real-world deployment. To demonstrate the resilience of heterogeneous policies, we deploy Scenario B (Sec. 6.3) to a real-world setting. The setup of the task is shown in Fig. 9b and is the same as in simulation. We use two holonomic RoboMaster S1 ground robots [16] (Fig. 9a), each running a customized model-based controller onboard [51]. We perform 10 runs for the trained HetGPPO and GPPO models both in simulation (Fig. 9c) and in the real world (Fig. 9d). As already discussed in Sec. 6.3, both the heterogeneous and the homogeneous models are able to solve the scenario in simulation, with the homogeneous model leveraging *inferred behavioral typing for physically identical agents*. This is shown in Fig. 9c, where all the runs of both models reach 100% task completion within 15s. On the other hand, as seen in Fig. 9d, the performance of the homogeneous model is heavily impacted in the real world. This is because, in this symmetric scenario, the homogeneous model cannot type agents based on position only, and has to rely on velocity observations to build the behavioral types. In practice, however, real-world estimated velocities can be noisy due to factors such as control/process delays and variability in the robot's measurement and motion models. Thus, relying on these observations makes the homogeneous (memory-less) model susceptible to erroneously switching the behavioral types dynamically (i.e., failing to distinguish if the robots are currently moving towards or away from the center). This leads to the plotted rollouts, where robots alternate the role of giving way to each other near the passage. Out of 10 runs, only 5 are completed within 60s. The heterogeneous model, on the other hand, does not rely on behavioral typing and is not impacted by the deployment noises, performing as well as in simulations.

8 CONCLUSION

In this paper, we introduced a new paradigm for learning heterogeneous policies in MARL. We motivated it with a categorization of techniques that homogeneous models can use to emulate heterogeneous behavior and empirically demonstrated their limits. Finally, we showed the benefits of policy heterogeneity for both performance and resilience on multi-robot tasks in simulation and in the real world. While we do not employ any methods to control the degree of heterogeneity of the agents' policies, we observe that training is already a good heterogeneity regularizer. In other words, if the system has heterogeneous requirements, HetGPPO will be able to learn them, while, if the system benefits from homogeneous policies, HetGPPO will learn the same policy as GPPO (with some loss in sample efficiency). In future work, we are interested in developing mechanisms that measure and actively tune the degree of policy heterogeneity in the team, allowing us to control the trade-offs between sample efficiency (of homogeneous policies) and resilience (of heterogeneous policies).

ACKNOWLEDGMENTS

This work was supported by ARL DCIST CRA W911NF-17-2-0181, the European Research Council (ERC) Project 949940 (gAIa), and in part by a gift from Arm.

REFERENCES

[1] Nora Ayanian. 2019. Dart: Diversity-enhanced autonomy in robot teams. *The International Journal of Robotics Research* 38, 12-13 (2019), 1329–1337.

[2] Tucker Balch. 2000. Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous robots* 8, 3 (2000), 209–238.

[3] Tucker Balch et al. 1997. Learning roles: Behavioral diversity in robot teams. In *AAAI Workshop on Multiagent Learning*.

[4] Spring Berman, Adám Halász, Vijay Kumar, and Stephen Pratt. 2007. Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2318–2323.

[5] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[6] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. 2022. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems* (2022).

[7] Jan Blumenkamp and Amanda Prorok. 2021. The Emergence of Adversarial Communication in Multi-Agent Reinforcement Learning. In *Conference on Robot Learning*. PMLR, 1394–1414.

[8] Elizabeth R Boroson and Nora Ayanian. 2019. 3D keypoint repeatability for heterogeneous multi-robot SLAM. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 6337–6343.

[9] Olli Bräysy and Michel Gendreau. 2005. Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation science* 39, 1 (2005), 119–139.

[10] Praneel Chand and Dale A Carnegie. 2013. Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. *Robotics and autonomous Systems* 61, 6 (2013), 565–579.

[11] Li Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. 2021. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021).

[12] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. 2021. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*. PMLR, 1989–1998.

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

[14] Mark Debord, Wolfgang Hönig, and Nora Ayanian. 2018. Trajectory planning for heterogeneous robot teams. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7924–7931.

[15] Ankur Deka and Katia Sycara. 2021. Natural Emergence of Heterogeneous Strategies in Artificially Intelligent Competitive Teams. In *Advances in Swarm Intelligence: 12th International Conference*. 13–25.

[16] DJI. Accessed: 2023-01-17. Robomaster S1. https://www.dji.com/robomaster-s1.

[17] Yousef Emam, Siddharth Mayya, Gennaro Notomista, Addison Bohannon, and Magnus Egerstedt. 2020. Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7719–7725.

[18] Yousef Emam, Gennaro Notomista, Paul Glotfelter, and Magnus Egerstedt. 2021. Data-Driven Adaptive Task Allocation for Heterogeneous Multi-Robot Teams Using Robust Control Barrier Functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9124–9130.

[19] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems* 29 (2016).

[20] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[21] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. 2020. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems* 33 (2020), 1970–1981.

[22] Brian P Gerkey and Maja J Mataric. 2002. Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 1. IEEE, 464–469.

[23] Dani Goldberg and Maja J Mataric. 1997. Interference as a tool for designing and evaluating multi-robot controllers. In *Aaai/iaai*. 637–642.

[24] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*. Springer, 66–83.

[25] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3040–3049.

[26] Chanyoung Ju and Hyoung Il Son. 2019. Modeling and control of heterogeneous agricultural field robots based on Ramadge–Wonham theory. *IEEE Robotics and Automation Letters* 5, 1 (2019), 48–55.

[27] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101, 1-2 (1998), 99–134.

[28] Soobum Kim, María Santos, Luis Guerrero-Bonilla, Anthony Yezzi, and Magnus Egerstedt. 2022. Coverage Control of Mobile Robots With Different Maximum Speeds for Time-Sensitive Applications. *IEEE Robotics and Automation Letters* 7, 2 (2022), 3001–3007.

[29] Ryan Kortvelesy and Amanda Prorok. 2022. QGNN: Value Function Factorisation with Graph Neural Networks. *arXiv preprint arXiv:2205.13005* (2022).

[30] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4501–4510.

[31] Ling Li, Alcherio Martinoli, and Yaser S Abu-Mostafa. 2004. Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior* 12, 3-4 (2004), 199–212.

[32] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3053–3062.

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

[34] Matthew Malencia, Sandeep Manjanna, M Ani Hsieh, George Pappas, and Vijay Kumar. 2022. Adaptive Sampling of Latent Phenomena using Heterogeneous Robot Teams (ASLaP-HR). *arXiv preprint arXiv:2208.06053* (2022).

[35] Sandeep Manjanna, Alberto Quattrini Li, Ryan N Smith, Ioannis Rekleitis, and Gregory Dudek. 2018. Heterogeneous multi-robot system for exploration and strategic water sampling. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 4873–4880.

[36] Siddharth Mayya, Diego S D'antonio, David Saldaña, and Vijay Kumar. 2021. Resilient task allocation in heterogeneous multi-robot systems. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1327–1334.

[37] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, et al. 2014. Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Field and service robotics*. Springer, 33–47.

[38] John F Nash Jr. 1950. Equilibrium points in n-person games. *Proceedings of the national academy of sciences* 36, 1 (1950), 48–49.

[39] Gennaro Notomista, Siddharth Mayya, Yousef Emam, Christopher Kroninger, Addison Bohannon, Seth Hutchinson, and Magnus Egerstedt. 2021. A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. *IEEE Transactions on Robotics* 38, 1 (2021), 159–179.

[40] Gennaro Notomista, Siddharth Mayya, Seth Hutchinson, and Magnus Egerstedt. 2019. An optimal task allocation strategy for heterogeneous multi-robot systems. In *2019 18th European Control Conference (ECC)*. IEEE, 2071–2076.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[42] Luciano CA Pimenta, Vijay Kumar, Renato C Mesquita, and Guilherme AS Pereira. 2008. Sensing and coverage for a network of heterogeneous robots. In *2008 47th IEEE conference on decision and control*. IEEE, 3947–3952.

[43] Amanda Prorok, M Ani Hsieh, and Vijay Kumar. 2017. The impact of diversity on optimal control policies for heterogeneous robot swarms. *IEEE Transactions on Robotics* 33, 2 (2017), 346–358.

[44] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.

[45] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2186–2188.

[46] María Santos, Yancy Diaz-Mercado, and Magnus Egerstedt. 2018. Coverage control for multirobot teams with heterogeneous sensing capabilities. *IEEE Robotics and Automation Letters* 3, 2 (2018), 919–925.

[47] Miguel Schneider-Fontan and Maja J Mataric. 1998. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation* 14, 5 (1998), 815–822.

[48] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[49] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Daniel Martin, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. 2022. Learning efficient diverse communication for cooperative heterogeneous teaming. In *Proceedings of the 21st*

International Conference on Autonomous Agents and Multiagent Systems. 1173–1182.

[50] Beining Shang, Richard Crowder, and Klaus-Peter Zauner. 2014. Swarm behavioral sorting based on robotic hardware variation. In *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications (SIMULTECH).* IEEE, 631–636.

[51] Ajay Shankar, Sebastian Elbaum, and Carrick Detweiler. 2021. Freyja: A full multirotor system for agile & precise outdoor flights. In *2021 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 217–223.

[52] Riccardo Spica, Eric Cristofalo, Zijian Wang, Eduardo Montijano, and Mac Schwager. 2020. A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Transactions on Robotics* 36, 5 (2020), 1389–1403.

[53] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. *Advances in neural information processing systems* 29 (2016).

[54] Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. 2020. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625* (2020).

[55] Ceyer Wakilpoor, Patrick J Martin, Carrie Rebhuhn, and Amanda Vu. 2020. Heterogeneous multi-agent reinforcement learning for unknown environment mapping. *arXiv preprint arXiv:2010.02663* (2020).

[56] Mingyu Wang, Zijian Wang, John Talbot, J Christian Gerdes, and Mac Schwager. 2021. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics* 37, 4 (2021), 1313–1325.

[57] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *International Conference on Machine Learning.* PMLR, 9876–9886.

[58] T Wang, T Gupta, B Peng, A Mahajan, S Whiteson, and C Zhang. 2021. RODE: learning roles to decompose multi- agent tasks. In *Proceedings of the International Conference on Learning Representations.*

[59] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations.*

[60] Javier Yu, Joseph A Vincent, and Mac Schwager. 2022. DiNNO: Distributed Neural Network Optimization for Multi-Robot Collaborative Learning. *IEEE Robotics and Automation Letters* 7, 2 (2022), 1896–1903.

[61] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

A EXPERIMENTAL SETUP

A.1 Simulation

We attach all the code used for simulations and training. Simulations are performed in the VMAS [6] simulator. All environments are customly created apart from Scenario B which is adapted from one of the scenarios already available in the simulator. The training is performed in RLlib [32] using PyTorch [41] and an implementation of the PPO algorithm for multi-agent training. The general training parameters used are shown in Tab. 2. Small variations of these are done on a per-environment basis and can be seen in the training scripts attached. The GPPO and HetGPPO model implementations and details are available in the code. Training is performed on a NVIDIA GeForce RTX 2080 Ti GPU. Each worker collects experience from the simulator using an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz.

Table 2: PPO training parameters.

Training		PPO	
Batch size	60000	ϵ	0.2
Minibatch size	4096	γ	0.99
SDG Iterations	40	λ	0.9
# Workers	5	Entropy coeff	0
# Envs per worker	50	KL coeff	0.01
Learning rate	5e-5	KL target	0.01

A.2 Real-world

Real-world experiments are performed using an Optitrack[7] motion capture system with 12 cameras to provide positional information to the robots. The robots used are holonomic RoboMaster S1 ground robots[8], running a customized model-based controller onboard [51].

B SCENARIO B REWARD STRUCTURE

The reward used to train Scenario B is comprised of two components: a positional reward and a collision reward.

The positional reward is proportional to the time delta in relative distance of an agent from its goal. In other words, a positive reward is assigned if an agent moves towards its goal and a negative one if it moves away. The agents receive a shared positional reward equal to the sum of their individual positional rewards. When both agents are placed on their goal, they keep receiving an additional final reward. The episode ends after 500 timesteps.

The collision reward is a constant penalty assigned to each agent in the presence of collisions. When training starts, the only collisions penalized are inter-agent ones. A curriculum is set up throughout training so that, when the agents' positional reward gets high enough to symbolize that they solved the task, collisions at the recesses start being penalized as well. This is done so that the agents are able to first learn to solve the task and can then fine-tune their performance by removing collisions.

C SIM TO REAL TRANSFER

To deploy policies trained in the VMAS simulator to the real world, we iteratively tune some simulation hyperparameters to fit the real-world conditions. These parameters are dependent just on the robots and their interaction with the real-world. Once tuned, they can be used for any training scenario.

The parameters that were key to a successfully deployment are linear friction and drag. Since we operate with ground robots at relatively low speeds, we set drag to 0 and tune linear friction. Through 3 real to sim iterations of binary search we were able to find the correct friction value for our robot-ground pair. Together with friction, we tuned the maximum acceleration in simulation to fit the real robot one. These parameters were tested and validated on simple single-robot tasks such as trajectory following and moving to a goal position.

[7] https://optitrack.com/

[8] https://www.dji.com/uk/robomaster-s1