Evaluating Benefits of Heterogeneity in Constrained Multi-Agent Learning

Alexandra Shaw

Supervisors: Prof. A. Prorok
M. Bettini

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Clare College August 2023

I would like to dedicate this thesis to my family.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

This project primarily builds upon and extends software from the HetGPPO repository and the VMAS multi-agent simulator (Bettini et al., 2022) from Prorok Lab. This project also utilizes the Monotonenorm package for the implementation of GroupSort and for a starting implementation of the 1-Lipschitz layer norm (Kitouni et al., 2021). Standard python libraries (PyTorch, NumPy, Scikit-Learn, Pandas, and Seaborn) were also used. Finally, the repository which contains all code written for this project is on Github: github.com/alexandrashaw10/mlmi-thesis.

This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains **14,934** words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Alexandra Shaw
August 2023

Acknowledgements

I would like to sincerely thank my supervisors Prof. Amanda Prorok and Matteo Bettini for all of their support, time, and invaluable insights. I'm truly grateful for the opportunity to work with you both and enjoyed all of the interesting discussions that we had throughout this project.

I would also like to thank my family, siblings, and friends for their constant support this year at Cambridge.

Abstract

In nature, many species achieve complex tasks even with individually limited cognitive capacity. Through diversity among members of the species, they can take on different roles and contribute to their overall survival. Inspired by natural systems, we are interested in understanding how multiple roles can be learned in multi-robot systems through diverse behavior between agents. Within Multi-Agent Reinforcement Learning (MARL), many multi-robot systems have used homogeneous policies that share the same policy for all agents. However, the limitations of this method are not yet properly understood. Heterogeneous policies give a separate policy to each agent, which increases the number of policy parameters proportional to the number of agents. Despite this drawback, they have recently been proposed as an alternative.

Within this work, we show how limiting the neural capacity of each individual agent in heterogeneous policies can increase diversity between agents, improve optimization, and provide robustness to input perturbation. Through varying neural capacity constraints, constraints from the environment, and scenario requirements, we additionally outline the fundamental issues of using homogeneous policies to learn multiple roles among agents in cooperative multi-agent settings.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

In biological collective systems, it is common to observe complex collective behavior arise from groups of individuals performing simple but diverse behaviors. For example, colonies of ants provide three prime examples. Armies of ants walk in organized lines to efficiently transport food. Colonies of ants can cooperate to survive otherwise catastrophic floods by creating a raft with their bodies by clinging to each others legs. Finally, ants with varying body sizes specialize into roles: smaller ants focus on harvesting while larger, stronger ants stand guard or clear large debris.

In each of these situations, the ant colony benefits from having more individual ants working together. They each require complex high-level behaviors: each ant must act according to its distinct role in the situation, and the ant must choose a role with an awareness of the behaviour of the other ants.

Recent research in robotics has been inspired by these biological collective systems to have robots learn to work together to achieve collective goals. Although these robots may learn together, each of these robots has their own brain, or policy, which they use to respond independently to their own observations in the environment. Specifically, this work is interested in learning complex behaviors while constraining the policy complexity on each individual robot.

Multi-agent robotic problems are classified into three types: *coordination, cooperation, collaboration*. Each of these types of robot problems have an analogous problem in nature. In *coordination* problems we seek to minimize interference between agents. *Cooperative* robot problems involve a gain in performance through teamwork. Finally, *collaborative* problems include agents with more than one physical type (Prorok et al., 2021). For all three types, we are interested only in scenarios where the agents share a common goal, especially scenarios which require behavioral *diversity*.

Many previous works have sought to analyze the ability of agents to work together in collective systems when each individual agent is *cognitively limited*. The famous flocking agents known as “boids” can effectively simulate the behavior of flocking birds by defining a simple set of behaviors for each agent (Reynolds, 1998). Each agent is given just three simple rules: align to the average direction of nearby boids, move towards the center of mass of nearby boids, and move away from boids that are too close. Despite their simple policies, the flocks of boids beautifully coordinate their movements to appear graceful and life-like.

Although the Boids simulator can create lifelike cohesive behavior, each agent has same behavioral rules and they are very simplistic. For most multi-agent cooperative problems, it is infeasible to come up with a set of rules by hand for each agent. Instead, multi-agent reinforcement learning (MARL) uses neural networks to learn policies. This work focuses on applying the idea of neural simplicity in a setting with behaviorally diverse agents that use neural networks to approximate their policies.

Through limiting the Lipschitz constant of the policy network, neural simplicity is imposed on each agent. Through a series of experiments, this work shows how limiting the neural capacity of each agent can promote or regularize diversity between agents, improve performance, and increase robustness of the learned policy.

Additionally, this work explores how the features and specifications of each scenario affect the required neural capacity of each agent. We provide insights into the inherent abilities of homogeneous policies and heterogeneous policies as these specifications vary.

1.1 Contributions

The main contributions of this work can be summarized as follows:

1. A novel application of Lipschitz continuous neural networks to decrease neural capacity of agents in multi-agent cooperative policies.

2. A novel method for measuring diversity that more accurately captures diversity within homogeneous policies.
3. A study of how neural constraints on heterogeneous agents can naturally increase role diversity within multi-agent, multi-role cooperative tasks.
4. An overview of environment conditions or task requirements for which homogeneous policies perform poorly for multi-agent, multi-role cooperative tasks.

1.2 Overview

This thesis is structured as follows:

- **Chapter 2 - Background:** Discusses the multi-agent reinforcement learning framework, introduces terminology used to discuss role diversity in cooperative multi-agent settings, and outlines existing methods for measuring diversity in multi-agent systems.
- **Chapter 3 - Methodology:** Proposes limiting the Lipschitz constant of each policy as a method for constraining the neural capacity of each agent. Introduces a new diversity measure that is effective for measuring agent specialization in homogeneous policies.
- **Chapter 4 - Environment Conditions and Diversity:** Demonstrates how agent specialization increases when neural constraints are imposed within scenarios that limit the size of the observation space, specifically physical space and feature space.
- **Chapter 5 - Behavioral Diversity with Physical Differences:** Presents the interaction between physical differences between agents, neural expressivity constraints, and diversity.
- **Chapter 6 - Conclusions:** Final conclusions are drawn about when neural capacity constraints are beneficial to multi-agent cooperative scenarios. Summarizes insights about heterogeneous and homogeneous policies.

Chapter 2

Background

2.1 Multi-Agent Reinforcement Learning

The problems considered in this work are partially observable Markov Games (POMG) (Kaelbling et al., 1998; Shapley, 1953). POMG are an extension of single agent partially observable Markov Decision Processes (POMDP) to the multi-agent setting. In POMG, agents do not have full information about the other agents or the underlying state of the Markov Decision Process (MDP). Instead, they must infer it through observations. A POMDP is defined by

$$\left\langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{N}}, \{\sigma_i\}_{i\in\mathcal{N}}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{\mathcal{R}_i\}_{i\in\mathcal{N}}, \mathcal{T}, S_0, \gamma \right\rangle \tag{2.1}$$

where $\mathcal{N} = \{1, \dots n\}$ is the set of agents, $\mathcal{S}$ is the set of possible shared states, $\{\mathcal{O}_i\}_{i\in\mathcal{N}}$ is the per-agent observation space and $\{\mathcal{A}_i\}_{i\in\mathcal{N}}$ is the continuous action space for each agent. $\{\mathcal{R}_i\}_{i\in\mathcal{N}}$ is the reward function which can be per-agent or shared across agents. The reward function is a function of the previous state, the actions taken by the agents, and the next state.

$$\mathcal{R}_i : \mathcal{S} \times \{\mathcal{A}_i\}_{i\in\mathcal{N}} \times \mathcal{S} \rightarrow \mathbb{R} \; \forall i \in \mathcal{N} \tag{2.2}$$

The function from states to observations $\{\sigma_i\}_{i\in\mathcal{N}}$, $\sigma_i : \mathcal{S} \rightarrow \{\mathcal{O}_i\}_{i\in\mathcal{N}}$, determines what the agents observe given the environment state. The stochastic transition model

$$\mathcal{T} : \mathcal{S} \times \{\mathcal{A}_i\}_{i\in\mathcal{N}} \times \mathcal{S} \rightarrow [0, 1] \tag{2.3}$$

defines the probability of transitioning into the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and the agents' actions $\{a_i \in \mathcal{A}_i\}$ $\forall i \in \mathcal{N}$. S_0 is the distribution over initial states. The discount factor γ controls the relative importance of immediate rewards and future rewards.

Each agent acts according to a policy $\pi_\theta(o_i)$. The policy is parameterized by θ and outputs an action distribution a_i for each agent given its observation o_i. A rollout in a scenario is defined as an instance of the $\mathcal{N}$ agents acting in the environment according to their policies π_i starting from a sampled initial state $s \sim S_0$ for T total time steps. The output action distributions from each agent's policy $\pi_i(\cdot) = a_i \sim \mathcal{N}(\mu_i, \sigma_i)$ are parameterized as diagonal, uni-modal Gaussian distributions with mean μ_i and variance σ_i.

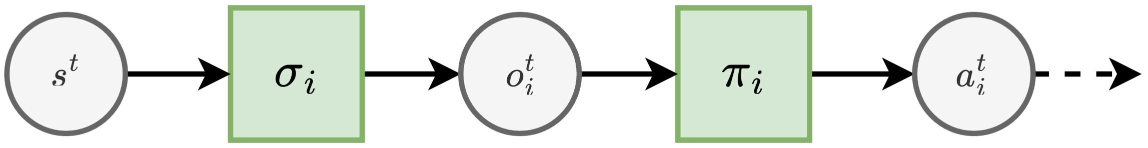


Fig. 2.1 The pipeline from state to action for a given agent.

During each time step t of a rollout, the underlying state s^t is passed into the observation function of each agent $\sigma_i(s^t)$ to output an observation o_i^t. Each agent inputs their observation o_i into their policy $\pi_i(o_i^t)$ to produce an action distribution a_i^t. Then an action $z_i^t \sim a_i^t$ is sampled from each agent and carried out. The new state s^{t+1} is updated according to the actions taken by each agent and the laws of physics. Given the new state, the actions of all agents, and the previous state, each agent receives its reward $r_i^t = R_i(s^t, \{a_i^t\}_{i \in \mathcal{N}}, s^{t+1})$. The processes, shown in fig. 2.1, is repeated until $t = T$.

Multi-Agent Reinforcement Learning (MARL) is a popular framework for solving POMG multi-agent robot scenarios. For all scenarios in this work, agents have a shared reward and work cooperatively towards the same goal. There are many algorithm types in MARL, but this project will investigate only model-free, Actor-Critic algorithms.

2.1.1 Training Multi-Agent Policies

Multi-Agent Proximal Policy Optimization (MAPPO) is a commonly used MARL algorithm for training in environments where agents have a common reward (Yu et al., 2022). It is an on-policy algorithm which utilizes a policy function, $\pi_\theta(o^t)$, and a value function, $V_\phi(s^t)$, both represented as neural networks. MAPPO is part of the family of Actor-Critic functions: the actor is the policy π_θ and the value function V_ϕ is the critic. The critic evaluates the value of each individual state, but it also provides an overall estimate of the optimality of the actor. Both the actor and the critic are optimized simultaneously using the PPO objective.

PPO is a policy gradient method which utilizes the objective:

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{2.4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t\ s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, so $r(\theta_{\text{old}}) = 1$ and $\hat{A}_t$ is an estimate of the advantage function at time t. This is an adjustment to the Trust Region Policy Optimization objective (TRPO), which is the first part of the PPO objective (eq. (2.4)). However, the second term reduces the maximum update that can be made to the policy by clipping the objective, reducing drastic changes to the policy.

Generally defining the action a_t as the collection of actions of all the agents, the advantage function is $A_t(s,a) = Q_t(s,a) - V(s)$ the difference between the Q-value for a state-action pair and the value function of the state. The Q-value is defined as $Q_t^\pi(s_t,a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}\left[\gamma^{t'-t}\mathcal{R}(s_{t'},a_{t'})|s_t,a_t\right]$, where $\mathcal{R}$ is the reward function. In other words, the Q-value of a state-action pair is the expected total sum of rewards after being in state s_t and taking action a_t. Rather than directly learning the Q-value function to evaluate the advantage function, PPO utilizes a technique called Generalized Advantage Estimator, or GAE(λ) (Schulman et al., 2016). This provides an estimate of the advantage function $\hat{A}_t$ at time t:

$$\hat{A}_t = R_t(\lambda) - V(s_t) \tag{2.5}$$

GAE utilizes the λ-return, which weights the average n-step returns with decay parameter λ (Sutton and Barto, 1998). The λ return is the infinite sum of returns of a given state. This can be made finite by considering the rewards up to a given time step T:

$$R_t(\lambda) = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1}\mathcal{R}_t^n + \lambda^{T-t-1}\mathcal{R}_t^{T-t} \tag{2.6}$$

λ can be used to balance the bias and variance of the value estimate, as changing λ to zero means that only single-step rewards are considered, while making $\lambda = 1$ gives the return of a full rollout.

MAPPO algorithm uses centralized-training decentralized-execution framework (CTDE), as the value network is only used during execution. The policy function outputs a stochastic action distribution which is shown to improve performance over deterministic actions. Due to its state-of-the-art performance on most tasks, MAPPO is used throughout this work.

The value of a shared policy is clear in a multi-agent setting. Utilizing a shared value function means that information about how an agent's individual performance affects the overall state is included in the policy gradients, as the loss utilizes the shared value function and advantage function (eq. (2.5)). However, it is not clear how to define the policy function when extending from single-agent reinforcement learning into a multi-agent domain.

Homogeneous Policies Many MARL solutions use *homogeneous* policies, where a single policy is shared by all agents. The training samples generated by each agent are pooled and used to update the shared policy. Most solutions use homogeneous policies (Hu et al., 2022), which can be quite limiting when learning diverse behaviors.

A common solution is to use an integer ID appended to the observations used to type each agent, but this can be a brittle solution and the number of behaviors that this policy can represent is still limited (Hu et al., 2022). Other works have also attempted to get around these constraints by increasing the complexity of the action distribution, for example by allowing the policy to learn multi-modal Gaussian action distributions (Ma et al., 2020). However, this technique is not a reliable method for training robust cooperative multi-agent policies because it relies on the randomness of each agents' actions to create diversity.

Heterogeneous Policies Alternatively, *heterogeneous* agents each have their own policy and they learn using only their own training examples. Heterogeneous agents do not share their policy parameters and the gradients generated from passing their training samples into the loss function are only back-propagated into their own policy. Importantly, the parameters of the value function of the heterogeneous agents are still shared. Although this method increases parameters and is less sample efficient, this can be a solution to increase the ability for agents to learn multiple behaviors (Bettini et al., 2023a). Employing heterogeneous policies can speed up and stabilize training and improve the optimal performance (Bettini et al., 2023a).

2.1.2 Learning Cooperative Behavior

Scenario, Goal and Environment Each multi-agent cooperative problem is called a *scenario* where the agents have an overall, shared *goal* within a certain *world*. Within a scenario, the *environment* is the world excluding the agents. The environment is defined by the spaces that the agent can move through, the objects in their vicinity, and some degree of intrinsic or aleatoric uncertainty. As with all reinforcement learning problems, the high-level goal is not the same as the reward function $\{\mathcal{R}_i\}_{i \in \mathcal{N}}$ which serves as a proxy to optimize for the true goal.

It is a well-known limitation of reinforcement learning that it is hard to define mathematical proxy reward functions for goals defined semantically. Poor choices of reward functions can prevent learning or cause unintended behavior (Hu et al., 2020; Skalse et al., 2022). Multi-agent settings involve learning the behaviors for multiple agents that act independently, which exacerbates the problem of choosing an appropriate reward function. Rather than attempting to define the actions of each agent individually, it is often easier to define an overall shared reward in cooperative scenarios.

Roles, Specialization and Diversity In each multi-agent scenario, agents must complete *tasks* according to their *role*. Tasks within a scenario are defined semantically as high-level sub-goals that must be completed to achieve the overall goal. Roles are also defined semantically as a kind of *specialization* of each agent, in the same way that a role would be described in a team of humans.

These terms best understood through an analogy to human behavior. For example, at a small startup, each member of the company will assume a different role based on their specialized skill set (e.g CEO and CTO). The members of the startup may have the same overall goal of getting funding. They will allocate tasks among themselves in order to reach that goal, such as the CTO designing their product prototype and the CEO designing the business strategy to present to investors. It is clear that these founders should have as much *diversity* in their roles and tasks as much as possible to utilize their limited man-hours most effectively. While some choices of how they divide their time are clearly better than others, there are often many ways that are best for how the founders of this company should allocate roles or choose tasks.

Motivated by the benefits of diversity in nature and in human teams, the concept of diversity is also extremely relevant in multi-agent systems. Relating multi-agent scenarios back to swarms of insects, reaching a certain level of diversity among the colony promoted their collective survival and ability to achieve complex tasks. This work seeks to increase the behavioral diversity between agents in multi-role scenarios. From the analogy of the two founders, it is clear that roles and tasks are a human-interpretable version of the actions that the agents take, are very scenario dependent, and agents can learn very different roles and tasks within a given scenario.

Successful Learning For a successful learning outcome in a cooperative problem, the chosen MARL algorithm must learn to allocate agents to complete *tasks* necessary for the overall goal. This requires learning (i) the optimal *roles*, (ii) the optimal *behavior* for each role, and (iii) the optimal *distribution* of agents into roles.

Without any limitations, the agents have full freedom to learn any number of roles for a given task. This motivates the study of how to shape the behavior, roles, and tasks learned by cooperative agents.

2.2 Diversity in Multi-Agent Systems

2.2.1 Existing Methods of Learning Diverse Roles

Recognizing the benefits of behavioral diversity outlined in section 2.1.2, there are many existing methods which increase diversity between agents. Grupen et al. (2022) utilizes interpretable deep learning techniques developed for medicine and presents a technique that forces agents to take on behaviors from a set of pre-defined, human-interpretable behaviors. While this strategy is useful in a high-risk environment like medicine, it is very limiting for robotic applications. Some effective methods enforce behavioral diversity among agents during training through changing the optimization objective, using an information theoretic approach Li et al. (2021) increase diversity between agents by maximizing the mutual information between the trajectories of agents during optimization. Wang et al. (2020) proposed a similar technique, but additionally assigned agents into roles using a stochastic embedding space parameterized as a multivariate Gaussian distribution. These works are effective at increasing the diversity among agents to a suitable level for the situation, but they require changing the optimization objective of the model and the neural capacity of each agent remains the same.

While these methods are useful, they do not limit each agents' individual capacity. Rather, this work is interested in understanding how the behavioral diversity among agents *naturally* changes when their neural capacity is limited.

2.2.2 Existing Diversity Measures

Measuring the diversity between agents is important to better understand the inter-agent interactions and unique contributions of each agents' actions in the world. Physical diversity, or $\mathcal{P}$-type diversity, can be measured by grouping robots based on their physical characteristics (Bettini et al., 2023a; Prorok et al., 2016). Behavioral diversity, or $\mathcal{B}$-type diversity, is much harder to describe mathematically as roles are determined through human interpretation as described in section 2.1.2.

Hierarchic Social Entropy (HSE) measures the behavioral difference between agents by discretizing the action space into a set of high level tasks and computing the amount of time that each agent spends per task (Balch, 2000). While the method used in HSE of summing the total difference in actions is useful, requiring discrete task definitions means that it is limited. It is also only suitable for deterministic actions.

A similar but more flexible diversity measure, Action-Based Role Difference (KL-RD) proposed by Hu et al. (2022) measures the symmetric KL-divergence between the action distributions of two agents at each time step (Hu et al., 2022). The measure sums the action distributions over a time window around each time step T to get a measure of the role difference r_i^T.

$$r_i^T = \frac{1}{2n+1} \sum_{t=T-n}^{T+n} \pi_i(o^t) \tag{2.7}$$

Computing the sum of action distributions around each point in time is important to capture similarities in actions that are periodic, which is important given that agents who follow the same policy will often produce the same actions but at different points in time. To compare two agents under this measure at a given time step T, the authors compute the symmetric KL-divergence between two role differences.

$$d^T_{\text{KL-RD}}(i,j) = KL(r_i^T \| r_j^T) + KL(r_j^T \| r_i^T) \tag{2.8}$$

This measure has a large drawback that the symmetric KL-divergence is not a distance measure. Under KL-RD the behavioral distance between agents in a policy does not vary in an intuitive manner and the distances between agents do not adhere to the triangle inequality, meaning that it cannot be used to accurately compare diversity. (Hu et al., 2022) also note that the action distribution alone does not capture the full measure of diversity between agents, as similar actions carried out in different world states can correspond to different roles. They add a trajectory-based distance measure which measures the total observation overlap between two agents. However, this measure is defined only for the specific observation type used in their experiments because it assumes each agent has local, circular observations.

System Neural Diversity (SND) created by Bettini et al. (2023b) defines a method for evaluating the neural diversity across heterogeneous agents. SND measures the uniqueness of actions for the policies given the same observations. It is defined as the pairwise Wasserstein distance between agents' action distribution outputs for all observations collected during a series of rollouts $\widehat{\mathcal{O}}$:

$$d_{\mathrm{SND}}(i,j) = \frac{1}{|\widehat{\mathcal{O}}|} \sum_{o \in \widehat{\mathcal{O}}} W_2(\pi_i(o), \pi_j(o)) \tag{2.9}$$

This approximates $\int_o W_2(\pi_i(o), \pi_j(o))do$ the total difference between action distributions output by the policy over the entire observation space. It uses the assumption that computing rollouts provides a representative sample of the distribution over observations. Using the Wasserstein distance — rather than the KL-divergence or symmetric KL divergence — is beneficial because it is a metric. The Wasserstein metric satisfies the triangle inequality, is symmetric, and does not go to infinity when the variance of the compared distributions approach zero. SND is a useful metric for evaluating overall diversity between heterogeneous agents, but it has some limitations. It makes the assumption that all agents' policies are well trained over all other agents' observation distributions, which may not be the case. Additionally, this method is zero for all agents who share parameters, as it measures the behavioral distance between two separate policies.

Chapter 3

Methodology

3.1 Neural Constraint Method

The primary goal of this work is to understand how limiting the neural capacity of each individual agent increases their ability to specialize into roles through the perspective of *function continuity*. Specifically, the neural capacity of each agent is defined by learning a policy from a class of functions with bounded rate of change called Lipschitz continuous functions.

3.1.1 Bounded Change in Neural Networks

A function $f : X \to Y$ is considered Lipschitz continuous if for every two real numbers $x, y \in X$, the distance between outputs is not more than a constant proportional to the distance between inputs. Consider two metric spaces (X, d_X) and (Y, d_Y) and a function $f : X \to Y$. If there is a $k \in \mathbb{R}^+$ such that:

$$d_Y(f(x), f(y)) \leq k\, d_X(x, y) \tag{3.1}$$

for all $x, y \in X$, then f is a Lipschitz function on X with Lipschitz constant k. Figure 3.1 provides a visualization of the Lipschitz constant of a 2D function.

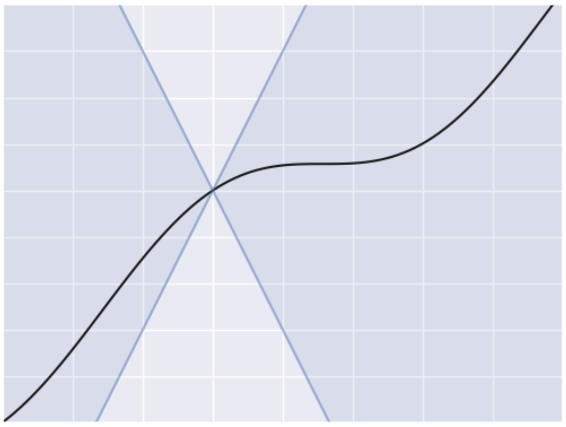


Fig. 3.1 An example of the Lipschitz constant of a function.

Lipschitz continuity applies across function compositions. Define functions $\{f_1, \dots, f_N\}$ that are Lipschitz continuous with corresponding constants $\{k_1, \dots, k_N\}$. Then their composition $F = f_1 \circ \cdots \circ f_N$ has a global Lipschitz constant K upper bounded by the product of the Lipschitz constant of each function (Ó Searcóid, 2006).

$$K = \prod_{i=1}^{N} k_i \tag{3.2}$$

Using this property, the maximum Lipschitz constant of the network can be set during training. Feed-forward neural networks (NN) are structured as a series of function compositions of layers and activation functions. The global Lipschitz constant of the network can be restricted by limiting the Lipschitz constant of each layer and activation function. The output of layer l, $y^{(l)}$, is the input $x^{(l+1)}$ of the next layer. If $\Phi^{(l)}$ is the activation function for layer l, the function $f^{(l)}$ for layer l is given by the affine transformation:

$$y^{(l+1)} = \Phi^{(l)}(W^{(l)}x^{(l)} + b^{(l)}) \tag{3.3}$$

By applying the definition, it is easy to show that the Lipschitz constant of a linear layer depends only on the norm of the weight matrix as the bias terms cancel out (Gouk et al., 2021).

$$\|(W^{(l)}x_1 + b^{(l)}) - (W^{(l)}x_2 + b^{(l)})\|_p \leq k\|x_1 - x_2\|_p \tag{3.4}$$

$$\|(W^{(l)}x_1 - W^{(l)}x_2\|_p \leq k\|x_1 - x_2\|_p \tag{3.5}$$

Finally, the Lipschitz constant of this layer is the supremum operator norm of W:

$$L(f^{(l)}) = \sup_{a \neq 0} \frac{\|W^{(l)} a\|_p}{\|a\|_p} \tag{3.6}$$

Then, the Lipschitz constant of a network with L layers denoting the Lipschitz constant of a function f as K_f is given by: as:

$$\prod_{l=0}^{L-1} K_\Phi^{(l)} \cdot K_W^{(l)} \tag{3.7}$$

If the activation for every layer $\Phi^{(l)}$ is chosen to be 1-Lipschitz ($K_\Phi^{(l)} = 1$), the global Lipschitz constant of the network is upper bounded by the Lipschitz constant of each layer $K_W^{(l)} = \|W^{(l)}\|_p$ (Scaman and Virmaux, 2018).

In this work, we use the method presented by Kitouni et al. (2021) for the layer normalization.

$$W^i \to W^{i'} = \frac{W^i}{\max(1, K^{-1/m} \cdot \|W^i\|_1)} \tag{3.8}$$

To set the global constant to K, each layer is divided by its matrix 1-norm, which normalizes it to be 1-Lipschitz. Then, each layer is scaled by $K^{-1/m}$ for a network of m layers to ensure the product of the Lipschitz constant of all layers is less than or equal to K. The maximum in the denominator is for numerical stability.

We constraint the 1-norm of the network, as was presented by Kitouni et al. (2021). While it is common to choose $p = 2$, corresponding to the spectral norm or largest singular value of W, computing it relies on an approximation through the power method. Instead, ℓ_1 utilizes the maximum absolute column sum norm and can be computed directly.

Each agents policy is constructed as a 3-layer MLP with the GroupSort activation function and weight normalization as described in eq. (3.8). Most popular activation functions are 1-Lipschitz: ReLU, Leaky ReLU, Softplus, Tanh, Sigmoid, and max-pooling but are shown to poorly approximate basic Lipschitz functions (such as the absolute value function) due to gradient attenuation. Alternatively, the *GroupSort* activation function is widely used for training Lipschitz continuous neural networks. It sorts the inputs from small to large in groupings, so it it is gradient norm preserving and 1-Lipschitz. It is proven to construct a universal Lipschitz approximator when used in a weight-normed neural network (Anil et al., 2018).

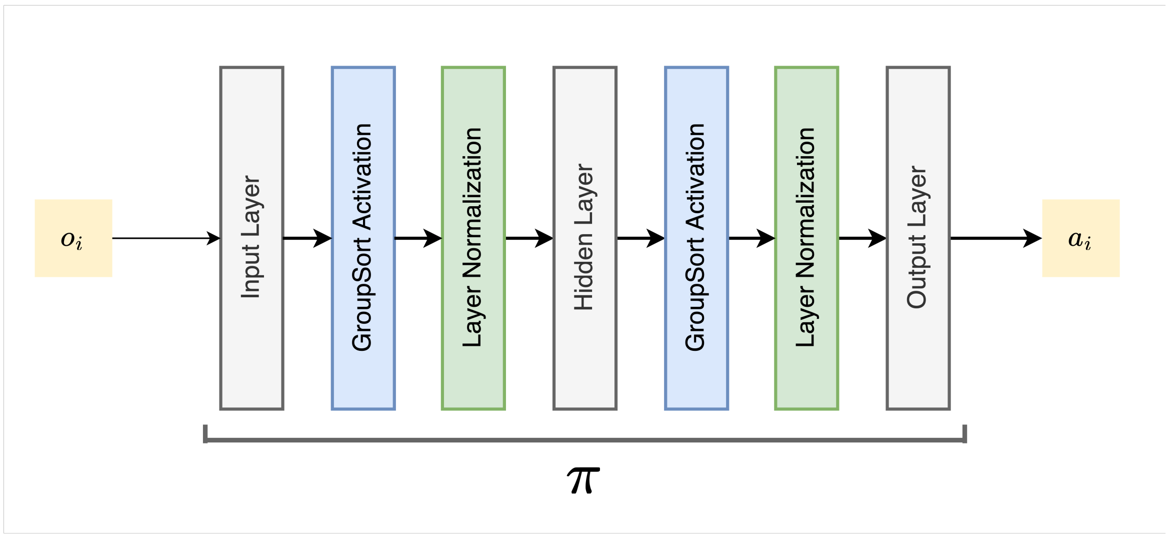


Fig. 3.2 A diagram of the policy network used in this work: a 3-layer MLP with GroupSort activation and layer normalization.

This constraint method allows for the freedom to tune the neural capacity of each agent to match the scenario. The goal of this project is to investigate ways in which complex collective intelligence can be achieved by agents that are cognitively limited by decreasing the Lipschitz constant of their policy.

3.1.2 Benefits of Lipschitz Continuous Neural Networks

Although there are numerous optimization benefits of Lipschitz continuous networks, the smoothing effects on the network is crucial for this project. This work utilizes the smoothing effects of constraining the global Lipschitz constant of the policy as a *tool* to naturally force simpler and smoother policies on each individual agent. The primary goal of this dissertation is to understand how this can increase behavioral diversity, cooperation, and the ability of agents to specialize into a role. Given the numerous additional benefits of Lipschitz continuous policies, this method is expected to improve performance, robustness, and most importantly diversity.

Many recent works have found that regularizing the Lipschitz constant of neural networks provides training stability and enhances optimization (Gogianu et al., 2021a; Gouk et al., 2021; Kitouni et al., 2021). Notably, Lipschitz continuous neural networks have enabled breakthroughs in Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). GANs optimize two neural networks simultaneously — a generator and a discriminator — with the goal of generating new samples that match a target distribution. The loss function rewards the generator for producing matching samples, while the discriminator is rewarded for correctly classifying samples as generated. When the discriminator is no longer able to distinguish between the original and generated samples learning is successful. Without regularization, optimizing the discriminator and the generator simultaneously often failed. When the discriminator was regularized by reducing the Lipschitz constant, training stabilized (Miyato et al., 2018). While the performance of Lipschitz continuous networks often improves, the benefits have mainly helped optimization by reducing the maximum gradients in the network (Gogianu et al., 2021b).

These findings were not within the field of reinforcement learning, but connections have been made between optimization for GANs and optimization for Actor-Critic methods (Pfau and Vinyals, 2016). Both model types require optimizing multiple networks simultaneously, where one network evaluates the performance of the other. This connection indicates that utilizing Lipschitz continuous networks for multi-agent reinforcement learning should also improve optimization of the policy and value function. Multi-agent actor-critic suffers from non-stationary training, as the actions learned by one agent affect the optimal actions of all other agents. Aside from optimization benefits, smoother action functions should also limit these issues. These optimization benefits are likely to be even more useful when considering heterogeneous agents with many policy networks.

Lipschitz continuous neural networks also provide loose robustness guarantees. Bounding the global Lipschitz constant of a network is a method of conservative certification. Around each input x, a network f with global Lipschitz constant K certifies that there is no δ such that $||f(x+\delta)-f(x)||_p \geq K||\delta||_p$ showing that the network is certifiably robust around a certain region around each input x (Cohen et al., 2019). By the definition of Lipschitz continuity, constraining the Lipschitz constant of a policy provides robustness to observation noise and smoothes actions across nearby observations.

Given all of these benefits, a few works have utilized Lipschitz continuous policies within single-agent reinforcement learning. Mysore et al. (2021) constrains the global Lipschitz constant of the network to smooth action outputs for drone flight. Kobayashi (2022) constrains local Lipschitz constants of the policy with a similar aim but constraining the local rather than global Lipschitz constant allows more policy expressivity and increases computational costs. These works found that decreasing the global Lipschitz constant caused over-smoothing, but the method used in these works did not allow for *tuning* the Lipschitz constant of the network as a hyperparameter.

The goal of this work is to investigate how constraining each agents' policy affects their ability to specialize into distinct roles. This involves limiting the *neural expressivity* of each agents' policy in a way that improves agent specialization. The method used to constrain the neural capacity of each agent investigated in this work is constraining the Lipschitz constant of the policy network.

We treat K as a hyperparameter and tune for the optimal K for a given model. This is done by searching for the lowest K such that the diversity and robustness increase but the policy performance does not decrease. $K = 1$ means the policy is isometric, and is a very strong constraint. K in the range $2-50$ per layer was found to be optimal for many networks (Gogianu et al., 2021a).

The terms neural capacity, neural expressivity, policy capacity and policy expressivity are all used to refer to the global Lipschitz constant set before training. This captures the essence that restricting the policy complexity of each agent is analogous to the limited brain capacity of each creature in natural collective systems. Each agents' policy is similar to a brain as it makes decisions for the agent based on the environmental stimulus it senses through observations.

3.2 Improved Diversity Measures

3.2.1 Wasserstein Diversity Metric

In order to accurately measure the diversity of agents in a multi-agent system, a measure that considers both action-based and a observation-based diversity must be considered. Given that roles and tasks are not mathematically defined, the concept of diversity between agents is defined in terms of outcomes. Agents are behaviorally diverse when they consistently perform different behaviors during rollouts. In this way, it is possible for homogeneous policies to learn specialized behavior. Actions alone don't translate to "different behaviors," as agents might take the same actions in different locations with completely different consequences. For example, a robot chef must know to put a cake in the oven and not the fridge, although

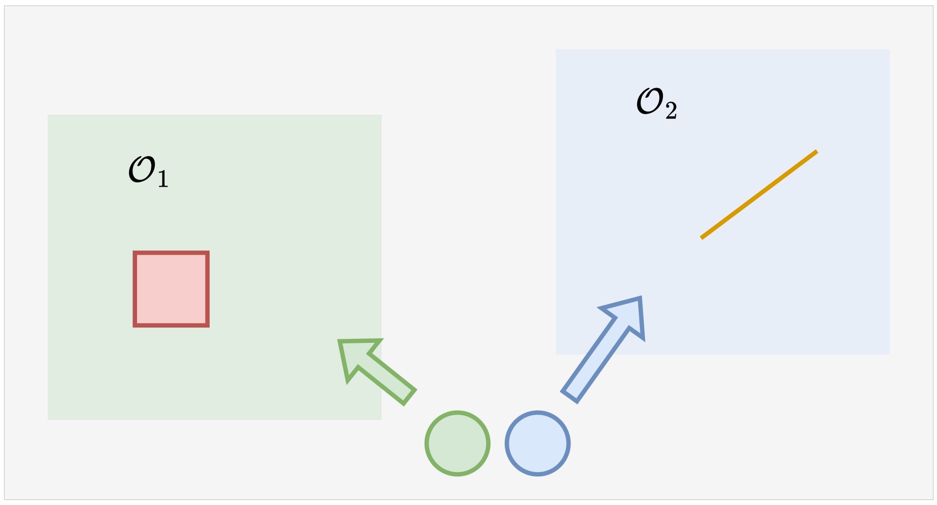


Fig. 3.3 Diagram of two agents that move into disjoint observation spaces $\mathcal{O}_1$ and $\mathcal{O}_2$, represented as physical spaces, during a rollout in order to complete two separate tasks.

both behaviors involve the same action sequence: grab a handle, open a door, and place the cake on a shelf.

While SND is suitable to measure heterogeneous diversity, an alternative metric is needed to measure the specialization of homogeneous agents. Under SND-defined diversity, homogeneous policies have zero diversity by design. The concept of diversity for heterogeneous agents, as measured by System Neural Diversity, is defined by integrating over the differences in action distributions for the same observations. This metric is used to evaluate the diversity of heterogeneous policies.

The concept of diversity for homogeneous agents can be partially defined by their ability to separate their observation distributions such that the actions that they take during rollouts are specialized. Although the agents use the same policy, separating their observation distributions during rollouts is one technique that they can exploit to increase the diversity of actions between agents to decrease functional constraints. In fig. 3.3, an example is shown which represents two homogeneous agents moving into disjoint observation spaces during a rollout in order to complete two separate tasks. The observation difference can be measured through the Wasserstein distance between the observation distributions of two agents.

$$d_{\text{obs}}(i,j) = W_2(\widehat{\mathcal{O}}_i, \widehat{\mathcal{O}}_j) \tag{3.9}$$

Similar to SND, a Monte-Carlo estimate of the observation distributions of the two agents can be collected through rollouts.

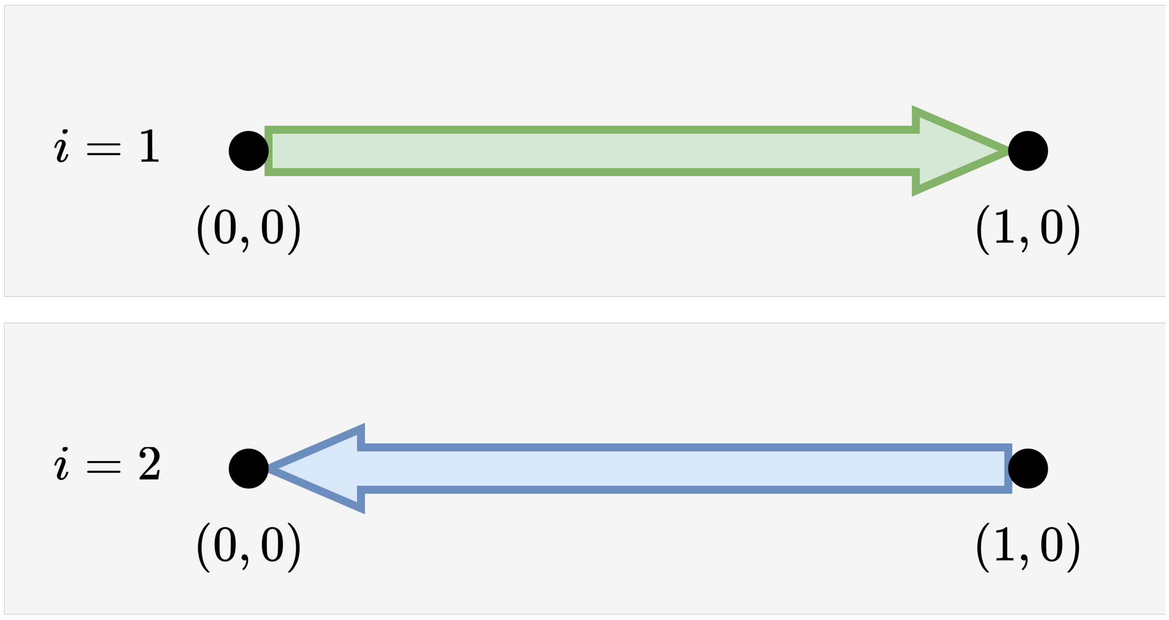


Fig. 3.4 A scenario depicting two agents moving between the same points at the same speed but in opposite directions.

However, considering each agent's distribution over observations is a poor measure of diversity when used alone. Consider the scenario represented in fig. 3.4. The agents have exactly the same observation distribution, but they clearly have unique behavior. However, the observation distribution between these two agents is zero. Therefore, the diversity measurement used must take into account both observation distribution and action distributions of the agents.

KL-RD (eq. (2.8)) is suitable for measuring the action diversity of a homogeneous policy by summing action distributions over a moving time window, but it uses the symmetric KL-divergence which is not a metric. Under this measure, it is not possible to directly compare two policies or even the measure at two different time steps. Additionally, the KL-divergence goes to infinity as the variance of the action distributions goes to zero, which is undesirable, as mentioned by Bettini et al. (2023b) when presenting SND. Instead, the Wasserstein distance metric is used in this work, making a metric diversity measure suitable for both homogeneous and heterogeneous policies.

Keeping the sum over action distributions proposed by Hu et al. (2022)

$$r_i^T = \frac{1}{2n+1} \sum_{t=T-n}^{T+n} \pi_j(o^t)$$

the distance measure can be updated to become the Wasserstein Role Diversity measure (Wass-RD):

$$d^T_{\text{Wass-RD}}(i,j) = W_2(r_i^T, r_j^T) \tag{3.10}$$

This modified measure is useful to understand the action distance between agents, but it is still limited by the fact that it does not consider the observation distance between agents. In order to present a measure of diversity that combines both, we present a new method in the next subsection that accomplishes this goal.

3.2.2 Ratio Measure of Diversity

We propose a new method for measuring diversity d_π that denotes the ratio between the action distribution and the observation distance between two observations for a single homogeneous policy π.

$$d_\pi(o_i,o_j) = \begin{cases} \dfrac{W_2\left(\pi(o_i),\pi(o_j)\right)}{\|o_i - o_j\|} & \text{if } o_i \neq o_j \\ \infty & \text{otherwise} \end{cases} \tag{3.11}$$

d_π is the Wass-RD action difference weighted by the distance between the two compared observations. While not perfectly scenario-agnostic, this ratio can provide a point of comparison between the behavioral diversity of policies in two different scenarios.

The ratio diversity as formulated measures the diversity between two observations. This measure can be used to diagnose the role diversity between two agents by computing the average d_π across all pairs of observations in a set of rollouts, averaged across a set of rollouts. Assuming total observations $|\mathcal{O}|$ per agent and pairs (o_1,o_2) such that $o_1 \in \mathcal{O}_1$ and $o_2 \in \mathcal{O}_2$:

$$\frac{1}{|\mathcal{O}|^2} \sum_{(o_1,o_2)} d_\pi(o_1,o_2) \tag{3.12}$$

The ratio diversity for agents i and j is the sum of the ratio diversity for all pairs of observations in the set of pairs of observations generated by agent i and agent j during rollouts.

This ratio can also be used to evaluate an entire policy by taking the maximum d_π over all pairs of observations such that the observations come from two different agents.

$$\max_{o_i,o_j \,:\, i\neq j} d_\pi(o_i,o_j) \tag{3.13}$$

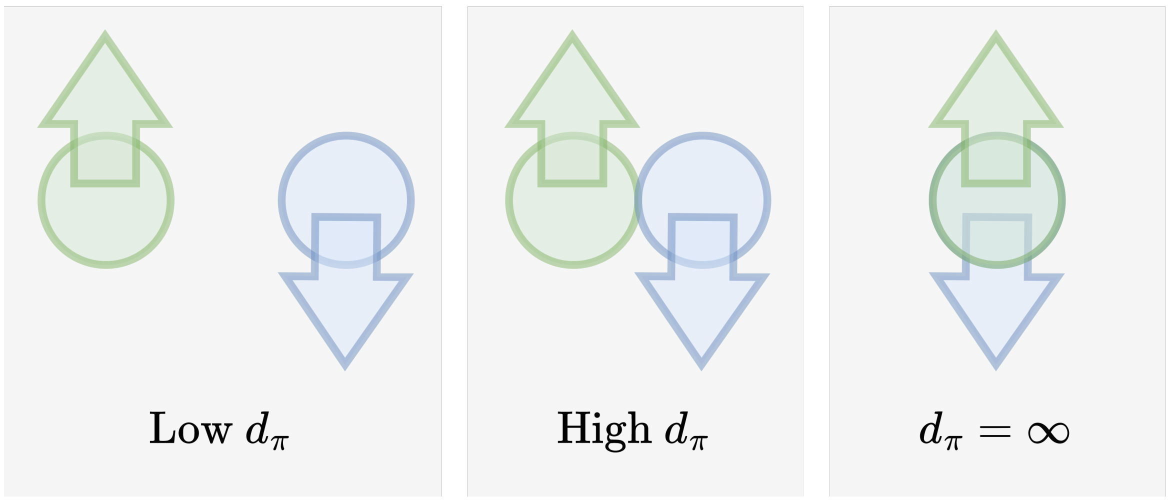


Fig. 3.5 Diagram showing examples of observations with low d_π, high d_π, and $d_\pi = \infty$.

Throughout this work, this ratio measure of diversity will be used to evaluate the diversity of homogeneous policies alongside the Wasserstein Role Diversity measure. For heterogeneous policies, the Wasserstein Role Diversity measure will be used to test diversity in action outcomes between agents and System Neural Diversity will be used to measure the policy difference between heterogeneous agents.

3.2.3 Connecting Diversity and Robustness

The ratio between the observation and action differences per agent for a homogeneous policy is an important indicator of the robustness of learning multiple roles given the task and environmental conditions. Taking the maximum d_π during a series of rollouts provides a lower bound for the Lipschitz constant of the policy defined by the Wasserstein metric and p-norm between the observations at a single time step. This ratio provides a secondary method for understanding the neural expressivity of a homogeneous policy, aside from the constant K set before training, which is suitable for both unconstrained and constrained policies.

In a noise-free system, a high ratio between action- and observation-difference can be beneficial to represent the optimal behavior for each observation exactly. However, a scenario that requires a high ratio within a homogeneous policy indicates a lack of robustness from the perspective of conservative certification and Lipschitz continuity. When this ratio is high, agents can accidentally "jump" from one behavior to another that is drastically different.

When d_π goes to infinity, it indicates an *impossibility* of role specialization between agents under π.[1]

A high ratio indicates that the Lipschitz constant, defined by the Wasserstein distance between action distributions and the p-norm used to measure observations, is high. This indicates a high risk of large behavioral shifts for each agent during a rollout, which can destabilize both training and executions.

This idea is the primary motivation behind constraining the neural capacity through constraining the Lipschitz constant of a policy. Tasks which require a high d_π to solve optimally should not be solved with a homogeneous policy and should instead be broken down into multiple tasks with lower d_π across heterogeneous policies. Although the Lipschitz constant constrained through the neural capacity restriction is not defined using the same distance measures, these two Lipschitz constants are proportional.

3.3 General Problem Formulation

This work will exclusively consider purely cooperative multi-agent robot scenarios. This means that all agents share the same goal and will have shared reward. Each robot has holonomic drive, meaning that they allow motion in any direction in the x-axis or y-axis without restriction. This facilitates focus on the high-level aspects of MARL. The action space $\mathcal{A}_i$ for each agent $i \in \mathcal{N}$ is a two-dimensional force vector

$$\mathcal{A}_i = \{\mathbf{u} := (u_x, u_y) \mid \mathbf{u}_i^{\min} \leq \mathbf{u} \leq \mathbf{u}_i^{\max}\} \tag{3.14}$$

where the individual force limits $\mathbf{u}_i^{\min}$ and $\mathbf{u}_i^{\max}$ represent the minimum and maximum output force for agent i motors.

Every agent is represented as a circular a ground robot with physical type denoted $\mathcal{P}_i = (m_i, r_i)$ determined by its mass m_i and body radius r_i.

The observations that the agents receive are their own positions and velocities as well as the relative positions and velocities of the other agent. For two agents, this is a good method for passing information between agents about their locations, although for this method to scale

[1]Notice that as observations are not evaluated based on time steps, agents do not need to be receive the same observations at the same time for d_π to reach infinity.

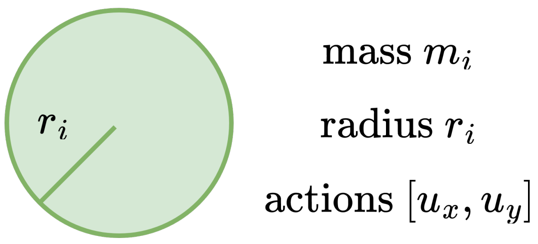


Fig. 3.6 Diagram representing the agents used in this work. Agents have physical type $\mathcal{P} = (m_i, r_i)$ determined by their mass m_i and radius r_i. Every agent has a two-dimensional action $a_i^t = [u_x^t, u_y^t]$ representing the force output in the x and y directions.

an alternative method for communication should be considered.

$$o_i = \left[p_x^{(i)}, p_y^{(i)}, v_x^{(i)}, v_y^{(i)}\right] \bigoplus_{j \neq i} \left[p_x^{(j)} - p_x^{(i)}, p_y^{(j)} - p_y^{(i)}, v_x^{(j)} - v_x^{(i)}, v_y^{(j)} - v_y^{(i)}\right] \tag{3.15}$$

Additionally, the addition of relative observations mean that the Lipschitz constant of the network for each policy depends on only the relative distance between agents, which still makes sense for this application. The details of this proof are shown in Appendix A.1.

3.4 Training Pipeline

The policy constraints are algorithm agnostic, and we can use any MARL training framework that we choose. Most experiments are trained with multi-agent PPO (MAPPO). The MAPPO implementation used for this work is modified from the implementation provided within the VMAS platform described below (Bettini et al., 2022). During each iteration, a batch of samples are collected with the current model parameters and stored in the replay buffer. For each epoch, the entire batch of collected data is used to train. To run each epoch, the samples are randomly divided into minibatches and sampled from the replay buffer. For each minibatch, the PPO-Clip loss is computed to generate gradients. The policy is optimized using Adam. Once the epochs are finished, the collector updates its copy of the model parameters with the updated model parameters. Then these parameters are used to collect a new batch of data from the environment. This repeats until training is complete.

the centralized critic and each agent's policy π_i are both represented by 3-layer multi-layer perceptrons (MLPs). More complex networks can be used, but the scenarios discussed are relatively simple and deeper networks are not needed. Homogeneous agents who share

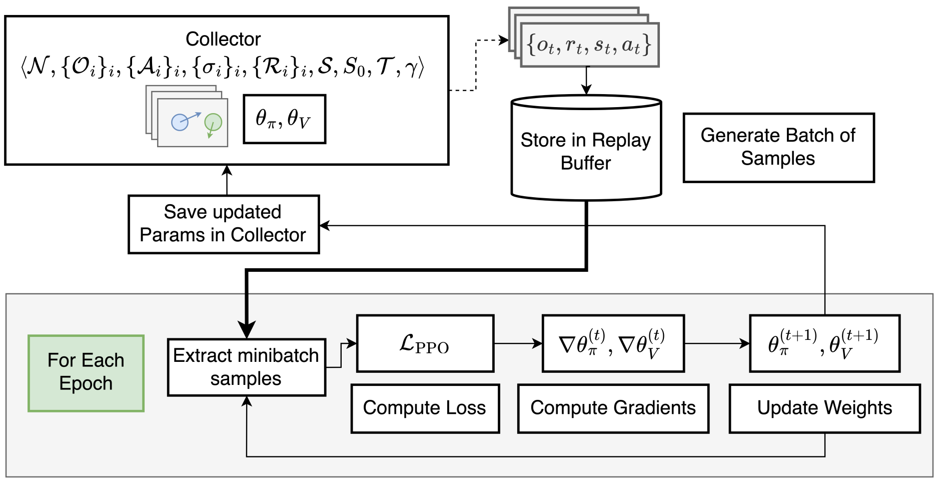

Collector
$\langle \mathcal{N}, \{\mathcal{O}_i\}_i, \{\mathcal{A}_i\}_i, \{\sigma_i\}_i, \{\mathcal{R}_i\}_i, \mathcal{S}, S_0, \mathcal{T}, \gamma \rangle$
θ_π, θ_V
$\{o_t, r_t, s_t, a_t\}$
Store in Replay Buffer
Generate Batch of Samples
Save updated Params in Collector
For Each Epoch
Extract minibatch samples
$\mathcal{L}_{\text{PPO}}$
Compute Loss
$\nabla\theta_\pi^{(t)}, \nabla\theta_V^{(t)}$
Compute Gradients
$\theta_\pi^{(t+1)}, \theta_V^{(t+1)}$
Update Weights

Fig. 3.7 An overview of the training process using the MAPPO algorithm on the VMAS platform.

a policy are represented by learning a single policy $\pi_i = \pi$ and each agent's observations are fed in as a batch dimension. Layer normalization is only included within the policy network and the centralized critic is not constrained. The value function is only involved in the optimization process, not during execution after training. Once the agents are trained, the critic is not used by the agents to make decisions. Therefore, is not considered part of their individual neural expressivity.

There are numerous multi-agent simulation platforms. Some multi-agent simulators are not designed specifically for robotics and use a videogame-like setting such as the commonly used StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). Others are designed specifically for robotics such as Multi-Agent Mujoco (de Witt et al., 2020), Gym-pybullet-drones (Panerati et al., 2021), and the Particle Robots Simulator (Shen et al., 2022). However, these simulators are not vectorized and are for specific applications.

For this work, experiments are trained using the Vectorized Multi-Agent Simulator (VMAS) developed by Bettini et al. (2022). VMAS has an underlying physics engine, each environment is fully customizable are there are a wide range of baseline scenarios. For algorithms like PPO which iteratively collect data and train in a supervised fashion, VMAS can reduce the sampling time by vectorizing the simulation process. Most environments completed sampling each iteration with 300 steps of 200 environments, or 60K training samples, in under 20 seconds. The environment can still have high computational cost to run the physics computations per step — such as computing distances between the objects in an environment with collisions — which can cause a bottleneck in sampling. For this reason, scenarios which require collision detection between more than a few objects are avoided.

This experimental setup is utilized for the remainder of the thesis. Now the key insights of this work are introduced in the coming chapters.

Chapter 4

Environment Conditions and Diversity

Within this chapter, we show that in scenarios that require multiple roles, learning many simple heterogeneous policies can be a considerably better solution for solving cooperative multi-agent tasks than learning one, complex homogeneous policy.

Further, we provide an understanding how neural expressivity requirements of the policy vary depending on environmental conditions, specifically limited feature space and observation noise. These insights can be used to diagnose when heterogeneous policies are beneficial over homogeneous policies.

4.1 Introducing the Left-Right Scenario

We begin with a simple scenario. The purpose of the Left-Right scenario is to show how increasing the space between agents decreases the required neural expressivity of the policy.

Treating the center of the world as $(x = 0, y = 0)$, the agents are spawned at distance d apart along the vertical line $x = 0$, with observation noise determined by $\sigma_i = \sigma$. The two agents each have a goal set before training at $(-1, 0)$ or $(1, 0)$. Both agents have the same physical type. Figure 4.1 provides a graphical depiction of the scenario and the scenario parameters are described in table 4.1.

The agents get reward proportional to their distance from goal at each time step. Agents are considered “on goal” when their body overlaps with the goal when the distance between their

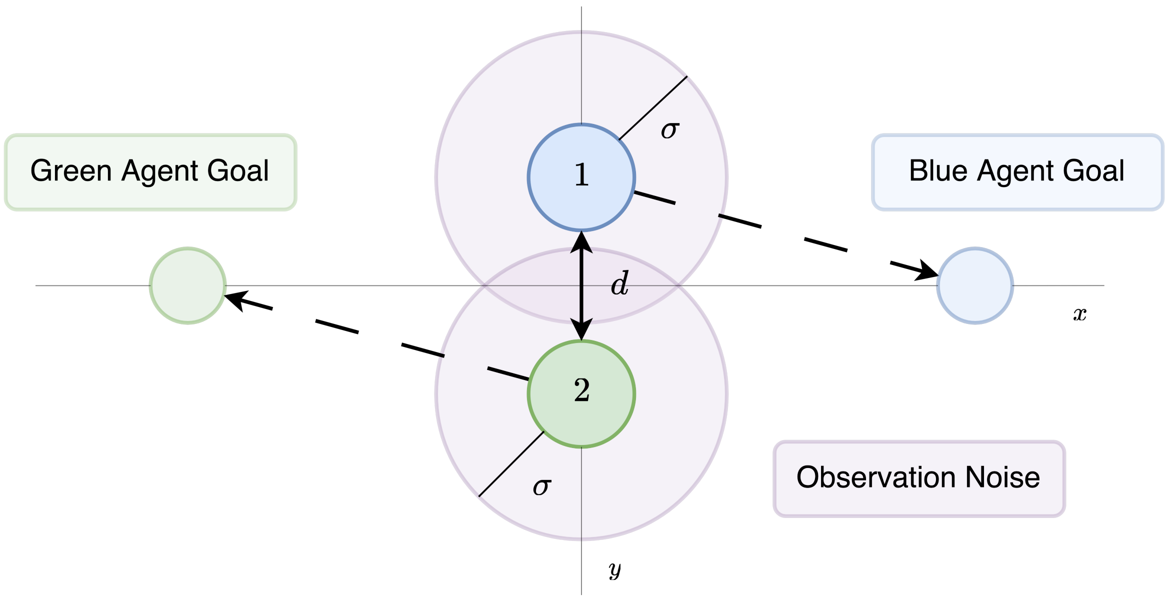


Fig. 4.1 A graphical overview of the left-right scenario.

centers is less than the sum of their radii. When both agents are on their respective goal, they get a final reward $R_g = 0.01$.

$$\mathcal{R}_{\text{Left-Right}} = \sum_{i \in \mathcal{N}} \|\vec{g}_i - \vec{p}_i\| + R_g \cdot \prod_{i \in \mathcal{N}} \mathbb{1}\left(\|\vec{g}_i - \vec{p}_i\| < r_g + r_i\right) \tag{4.1}$$

At the start of the scenario, the agents are spawned symmetrically along the vertical axis $x = 0$. The optimal policy corresponds to both agents moving directly towards their goal in opposite directions. To perform optimally, they must also slow down to a stop on the goal position and minimize oscillations over the goal. In this problem, none of the bodies can collide, so the agents can drive through each other.

Initially, both agents are spawned close to the origin at distance 2×10^{-2} apart. Despite the fact that this distance is less than the agent radius, both heterogeneous and homogeneous models can learn the optimal policy. However, learning is be more difficult for homogeneous policies. The training samples for the the two agents at the start of training will be overlapping, but the agents will optimize the policy in different directions.

Left Right Scenario Configuration	
Reward Type	Shared
Position Reward	$\sum_{i \in \mathcal{N}} \|g_i - p_i\|$
Goal Reward	$R_g = 0.01$
Goal Positions	$[(-1,0),(1,0)]$
Goal Characteristics	Sphere, radius $r_g = 0.05$
Collisions Possible	No
Num Agents	2
Agent Mass m_i	2.0
Agent Size r_i	0.15
Spawn Noise	0
Vertical Spawn Separation	0.02

Table 4.1 The configuration of the Left-Right scenario.

4.1.1 Training with Neural Constraints

With a close spawn distance, it is hard for the homogeneous agents to learn the optimal policy and reach their respective goals. The performance of the unconstrained homogeneous policy is much lower than the performance of the unconstrained heterogeneous policy, shown in fig. 4.3. Similarly, the training curves of the heterogeneous and homogeneous policies in fig. 4.2 show that the homogeneous agents take longer to train and have more variance across the three different models.

Furthermore, fig. 4.3 shows the effects of the neural capacity constraints on both the homogeneous and heterogeneous models. Aside from the optimization difficulty that this scenario presents for the homogeneous agents, the required behavior is simple. Through these performance plots, we see a benefit of limiting the neural expressivity within both homogeneous and heterogeneous models. The homogeneous model, although it takes more iterations to converge to the optimal policy (fig. 4.2), receives a substantial boost in performance with

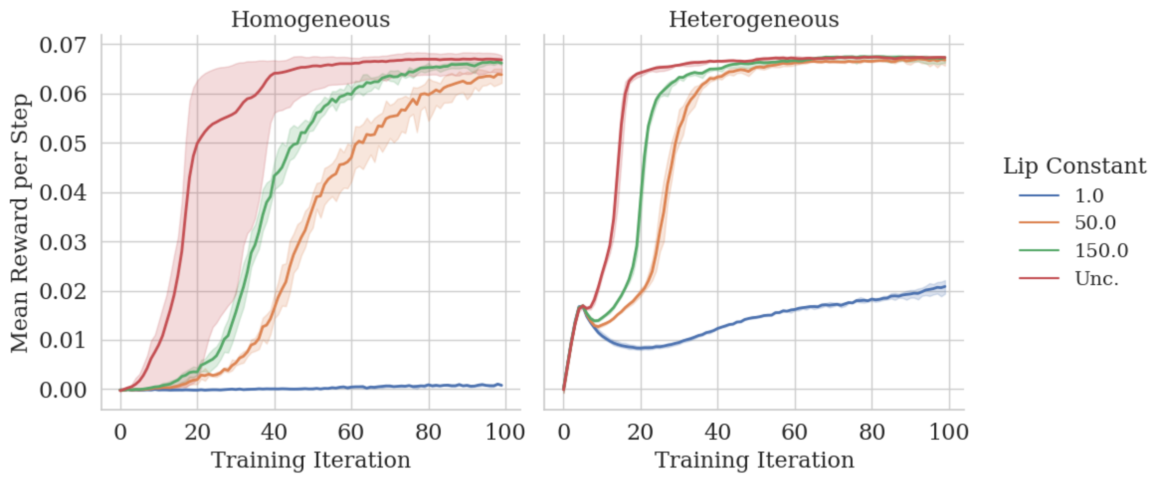


Fig. 4.2 Training curves showing the average reward per time step over 100 training iterations of heterogeneous and homogeneous policies in the Left-Right scenario with spawn distance 2×10^{-2}.

the neural constraints of $K = 50$ and $K = 150$ applied and the homogeneous policy almost reaches the same level of performance as the heterogeneous policy.

Through these performance and training plots, the benefit of heterogeneity in this scenario is clear. The training plots are much smoother and the policies converge more quickly. However, it is not yet clear whether the neural expressivity constraints have improved the heterogeneous policy, as the performance remains relatively unchanged. Through measuring the diversity and robustness to observation noise of the constrained and unconstrained policies, we can understand the benefits of limiting the policy expressivity.

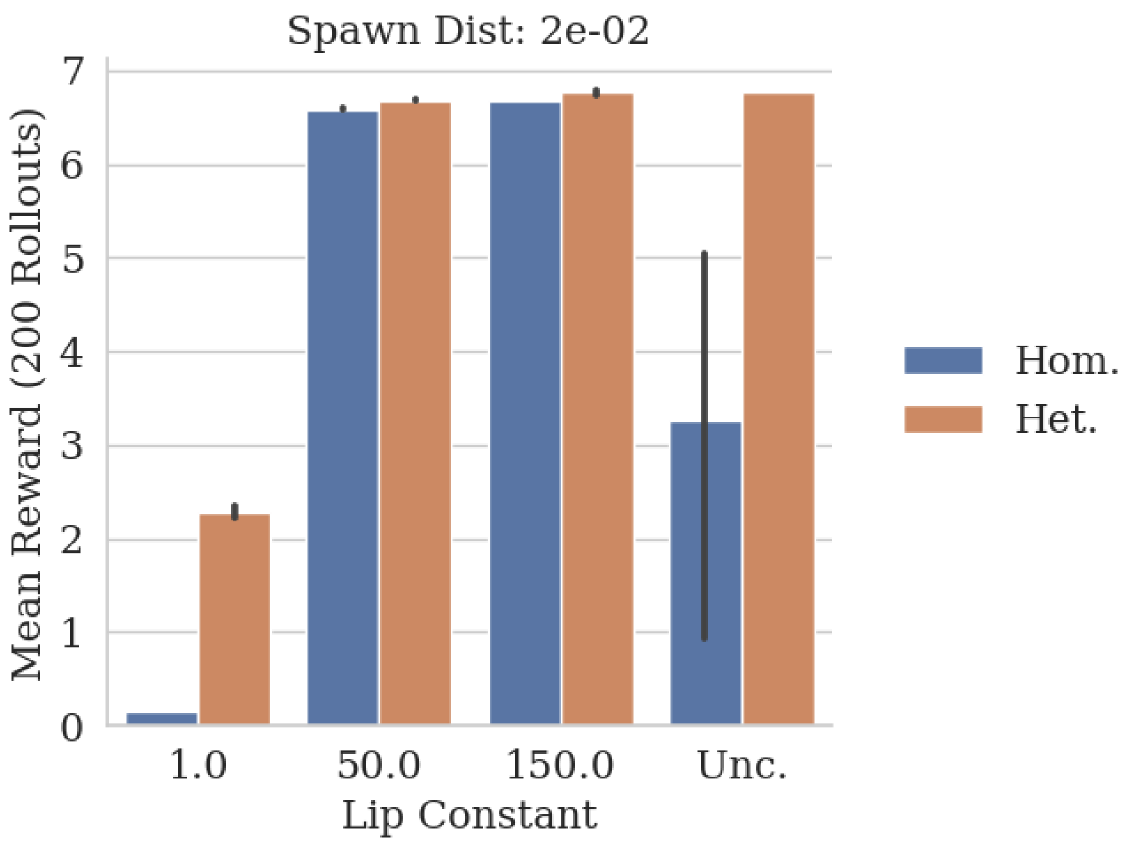
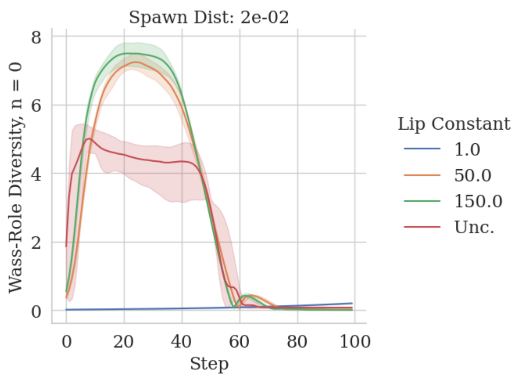
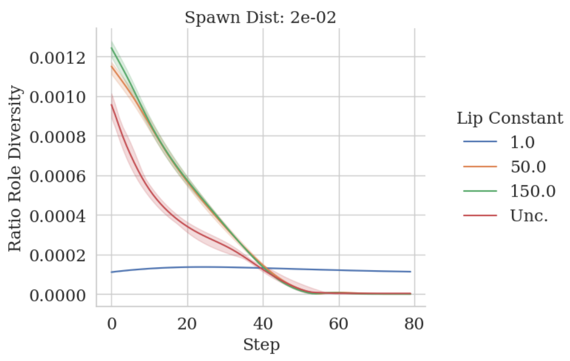


Fig. 4.3 The average total reward of homogeneous and heterogeneous policies in the Left-Right scenario starting with a spawn distance of 2×10^{-2}, averaged over 300 environments with 100 steps each over 200 rollouts.

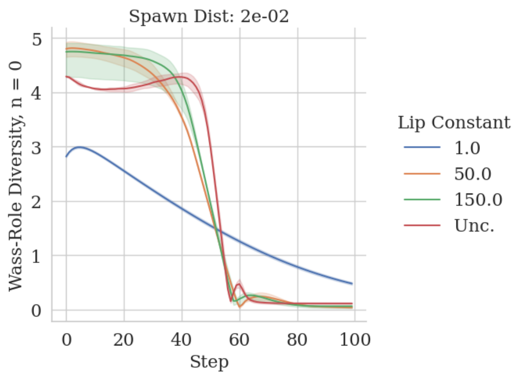
Diversity Increase under Neural Constraints



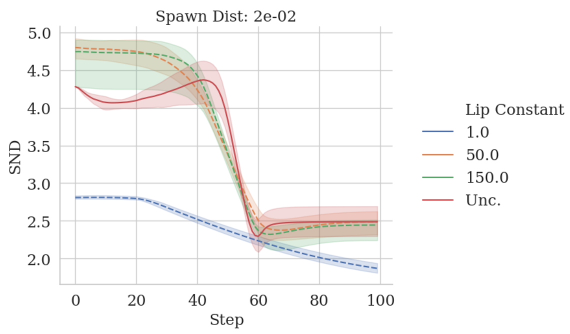
(a) Homogeneous Wass-RD



(b) Homogeneous Ratio diversity d_π



(c) Heterogeneous Wass-RD



(d) Heterogeneous SND

Fig. 4.4 Wass-RD for both policy types with varying neural constraints in the Left-Right scenario with spawn distance 2×10^{-2}, averaged over the steps of a rollout for different neural constraints over 200 rollouts with 300 environments each.

Under neural constraints the heterogeneous agents increase diversity both according to System Neural Diversity (SND) (fig. 4.4d) and Wasserstein-Role Difference (Wass-RD) (fig. 4.4c). There are only two agents and the actions are simple, so by considering the plot of the average Wass-RD throughout a rollout, we can separate the actions of the agents into stages fig. 4.4c. From steps 0 to 50, the agents immediately move in opposite directions, starting with action diversity of over 4. This action diversity remains relatively constant until they near the goal, when they slow down and the Wass-RD value decreases as their force output goes to zero. Then the agents apply a small force in the opposite direction to come to a stop on the goal. With neural constraints, the agents start with higher initial action diversity and the action difference changes more smoothly, reducing speed more gradually. Rather than immediately applying an opposite force, they slow down more gradually on the goal.

The homogeneous policies also increase in diversity, measured through the average Wass-RD (fig. 4.4a) and average ratio diversity (fig. 4.4b). From fig. 4.3, we know that the homogeneous unconstrained policy does not perform well. Similar stages of actions are visible in the unconstrained homogeneous policy to the heterogeneous unconstrained policy, but the initial action distribution between agents starts near zero and the agents have much more variance in their actions across different rollouts. Instead, under the neural constraints, the agents are able to learn a policy with artificially increased diversity which is much higher at its maximum peak than the heterogeneous policy. This shows how homogeneous policies can also take advantage of neural constraints to increase role specialization.

Finally, all plots show that with $K = 1$, both models exhibit over-smoothing and do not learn the optimal policy. Physically, it makes sense that a Lipschitz constant of 1 is too strong of a constraint for this scenario as it does not allow the agents to quickly decrease force output when they reach the goal. Additionally, the homogeneous policy is forced to output similar actions for both agents when they are at the starting position, removing role separation between agents. The metric diversity fig. 4.4b shows that the metric diversity throughout rollouts is constant and low and the Wass-RD shows action difference near zero. On the other hand, fig. 4.4c and fig. 4.4d show that the hom heterogeneous policies still display unique actions, albeit less diverse. This shows that the neural expressivity of the homogeneous policy must be higher than the heterogeneous policy for agents to specialize in this scenario.

4.2 Changing Environment Conditions

4.2.1 Modifying Left-Right: Spatial Constraints

Within the previous section, we discussed how neural capacity constraints can affect learning with homogeneous and heterogeneous agents. We found that both policies improved under neural constraints, but heterogeneous policies performed better and had higher diversity between agents. In this section, we vary the scenario parameters to first show scenarios where homogeneous policies may be a better choice. Then, we demonstrate when heterogeneous policies are *necessary* by showing scenarios where it is not possible for homogeneous policies to learn the optimal policy.

4.2.2 Increasing Spawn Separation

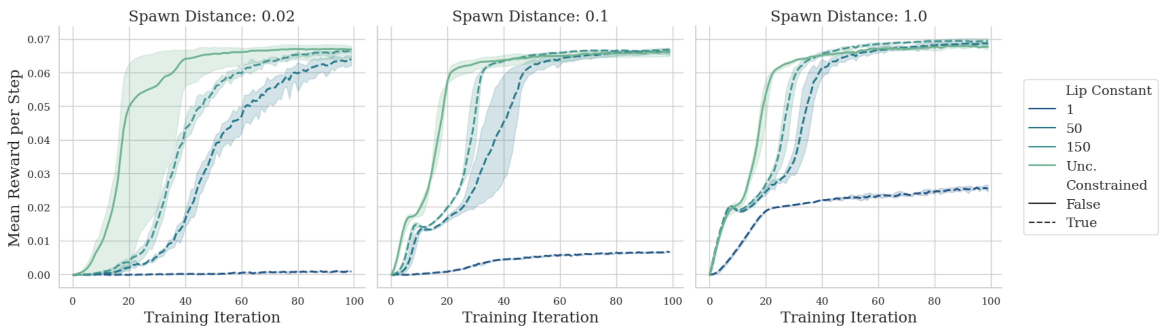
From the perspective of reinforcement learning, the increase in spawn distance makes the scenario harder as the agents have to travel farther to reach their goal. The formulation of the GAE (eq. (2.5)) uses an exponentially weighted sum of rewards to estimate the value of a given state. When it takes more time steps for agents to reach the final goal, the strength of the goal reward is limited by the exponential decay factor λ. Increasing the distance to goal while keeping the final goal reward fixed slightly decreases the reward signal at the start of the rollout.

Despite this, as the spawn distance between agents increases, the scenario becomes easier for homogeneous policies: as the agents are not required to interact at all and their observation regions can become disjoint. This allows the policies to learn diverse behavior because the agents are less limited by the functional constraints; the agents can optimize different regions of the policy input space to match the differing behavior. Figure 4.5 shows the training curves with the different spawn separations. The heterogeneous agents are relatively invariant to the starting position, with slight increases corresponding to the distance to goal, while homogeneous agents benefit from sharing parameters as the distance increases.

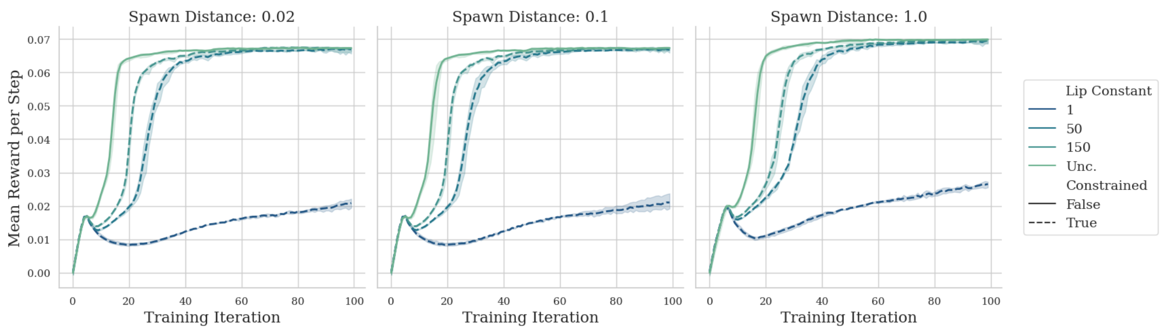
Figure 4.6 shows that the performance of the homogeneous policy increases as the spawn separation increases. However, the homogeneous policy does perform worse with no constraints at all distances, likely due to poor optimization. Heterogeneous policies benefit from neural constraints most when the spawn separation is increased, as the neural smoothness constraints help with generalization of the training samples. Specifically for $K = 1$, we can see how the homogeneous policy improves as the neural expressivity requirements of the scenario are relaxed.

4.2.3 Degenerate Case: Zero Spawn Distance

In the previous section, we discussed how as the spawn distance between agents decreases, the required neural expressivity must increase. Heterogeneous agents divide the required neural expressivity among different agents. For example, when two people try to remember a phone number, the chances that they remember the number correctly is massively increased if they each remember half rather than both memorizing all of the digits. Instead, homogeneous agents must contain all of the required neural expressivity in a single policy.



(a) Homogeneous policies.



(b) Heterogeneous policies.

Fig. 4.5 Training curves showing the average reward per time step over 100 training iterations of both policy types in the Left-Right scenario with increasing spawn distances.

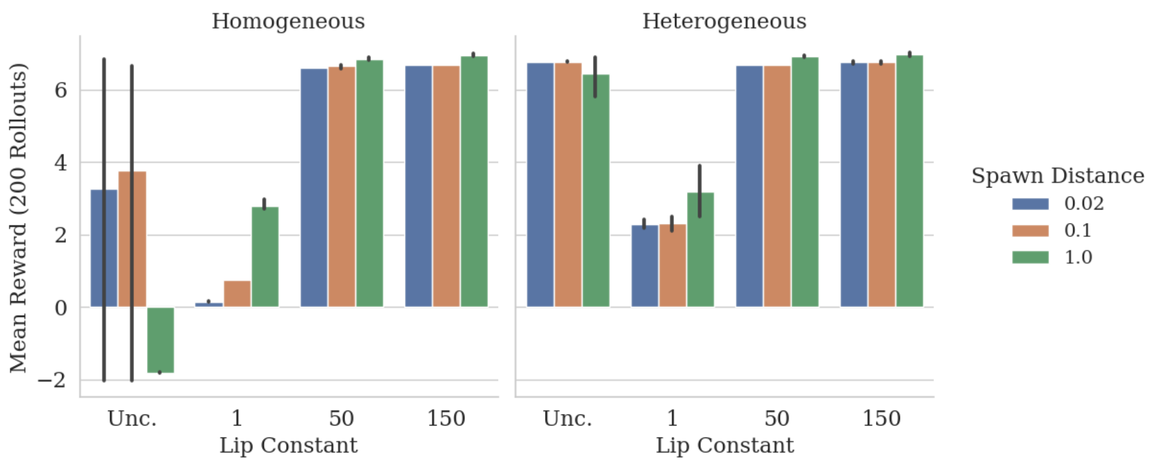


Fig. 4.6 Performance of both policy types showing the average total reward after 200 rollouts on 300 environments each in the Left-Right scenario with increasing spawn distances.

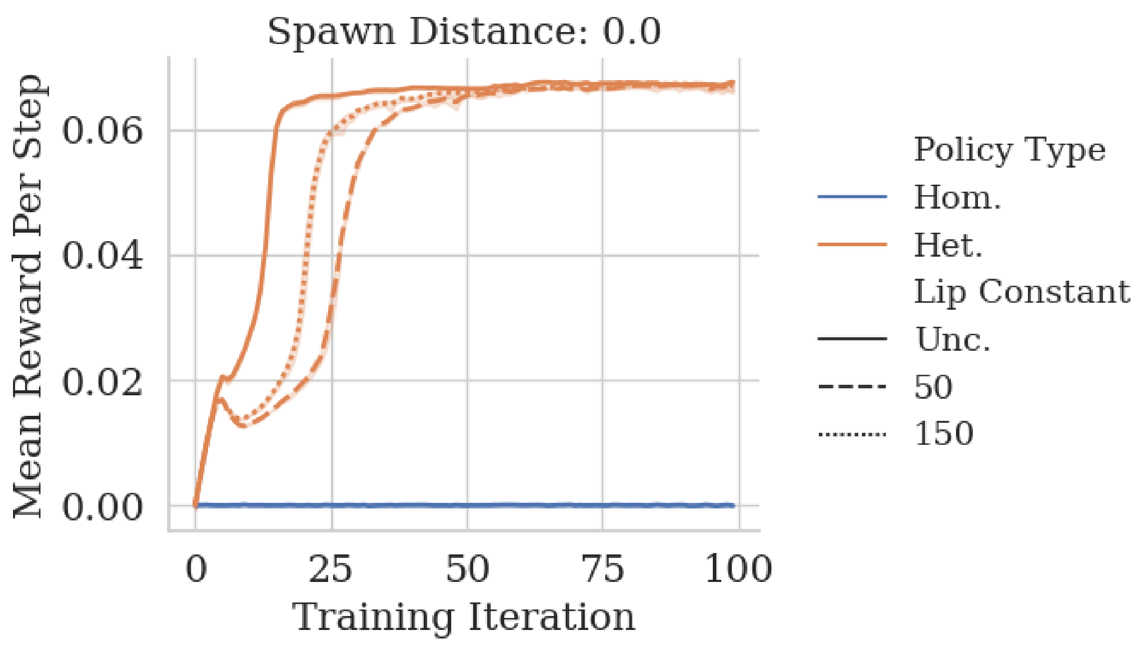


Fig. 4.7 Training curves showing the average reward per time step over 100 training iterations of both policy types in the Left-Right scenario with zero spawn distance.

In fig. 4.7, it is clear that with a spawn distance of $d = 0$, the homogeneous agents are not able to make meaningful actions when starting at the same position. The agents are physically the same and have the same sensor function σ, so each agent will always output the same action distribution at the start when there is no observation noise.

This shows a basic, but fundamental issue with homogeneous policies. Homogeneous policies are functions, so they can only ever output a single action distribution for each observation. This action distribution can be made multi-modal or high variance, but that requires that the agents rely on stochasticity to act according to their role, which is a very unreliable solution. Further, it is unlikely that the agents will be able to learn the optimal policy if they cannot reliably sample unique actions at the start of rollouts, preventing them from learning the policy in the rest of the space.

With zero spawn separation, the observations at time step $t = 0$ of both agents are the same: $o_1^t = o_2^t$. Therefore, the ratio diversity measure d_π goes to infinity. This represents the "impossibility" of learning a stable policy homogeneously that encodes multiple roles.

Alternatively, it is clear that heterogeneous agents can easily learn the optimal policy. Agents with the same observations can output independent action distributions, allowing heterogeneous agents to be much more expressive with the same neural capacity. This is a necessary tool to learn multiple unique roles. Additionally, the diversity ratio is considered for a single agents' policy only. As no two agents in the scenario act with the same policy, there is no longer an impossible condition that the agents must meet.

Through these results, we expose fundamental problems with using homogeneous policies to learn multi-agent cooperative policies in environments with limited space between agents.

4.3 Introducing the Give Way Scenario

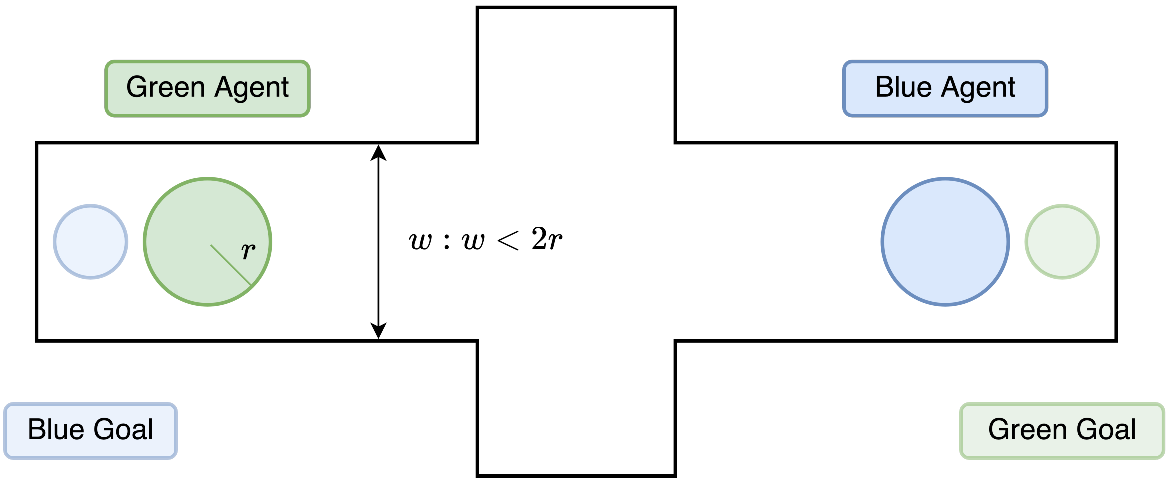


Fig. 4.8 A diagram depicting the Give Way scenario.

The "Give Way" Scenario depicted in fig. 4.8 is an extension of the Left-Right scenario. Now, the agents must move through a narrow corridor to reach their goal at the other end of the corridor. However, only a single agent can pass through the corridor at a time. Therefore, at least one agent must "give way" and move into the wider opening to let the other agent pass before reaching its goal.

This scenario extends the insights about spatial constraints and tests the neural capacity constraints in an environment which requires more complicated actions and behaviors than the Left-Right scenario. The scenario uses the same reward function $\mathcal{R}_{\text{Left-Right}}$ in eq. (4.1).

This scenario is very hard for homogeneous agents to solve, and often the agents reach the end of the corridor then turn around and begin to move back into the center of the corridor to give way once again. There is a slight performance improvement gained from the constraints

Give Way Scenario Configuration	
Position Reward	$\sum_{i \in \mathcal{N}} \lVert g_i - p_i \rVert$
Goal Reward	$R_g = 0.01$
Goal Positions	At the ends of the corridor
Goal Characteristics	Sphere, radius $r_g = 0.05$
Collisions Possible	Yes
Num Agents	2
Agent Mass	2.0
Agent Size r_i	0.15
Agent Actions a_i^t	u_x^t, u_y^t

Table 4.2 The configuration details of the Give Way scenario.

in the homogeneous policies. Heterogeneous agents can very easily solve this task for all neural constraint levels, shown in fig. 4.9.

4.3.1 Diversity with Limited Physical Space

The Wass-RD of each policy in fig. 4.10 shows that under neural constraints, the homogeneous models become less diverse, leading to the decline in performance visible in fig. 4.9. Meanwhile, the heterogeneity of all of the heterogeneous models increases with the constraint.

Through the Wass-RD plot, the three stages of the policy are once again visible. Steps 1 to 100 correspond to agents moving through first segment of the corridor. Both homogeneous and heterogeneous policies can complete this step. Then, around 100 steps, the heterogeneous policies demonstrate the “give way” behavior, which causes a drop in diversity while the agents slow down and utilize the larger section of the corridor. Finally, diversity increases again as the agents reach the end of their corridor and settle into a final resting position against the wall, sometimes bumping into the wall causing oscillations. In the homogeneous W-RD plot, these three stages are less clear. Without the constraint, after the third stage the

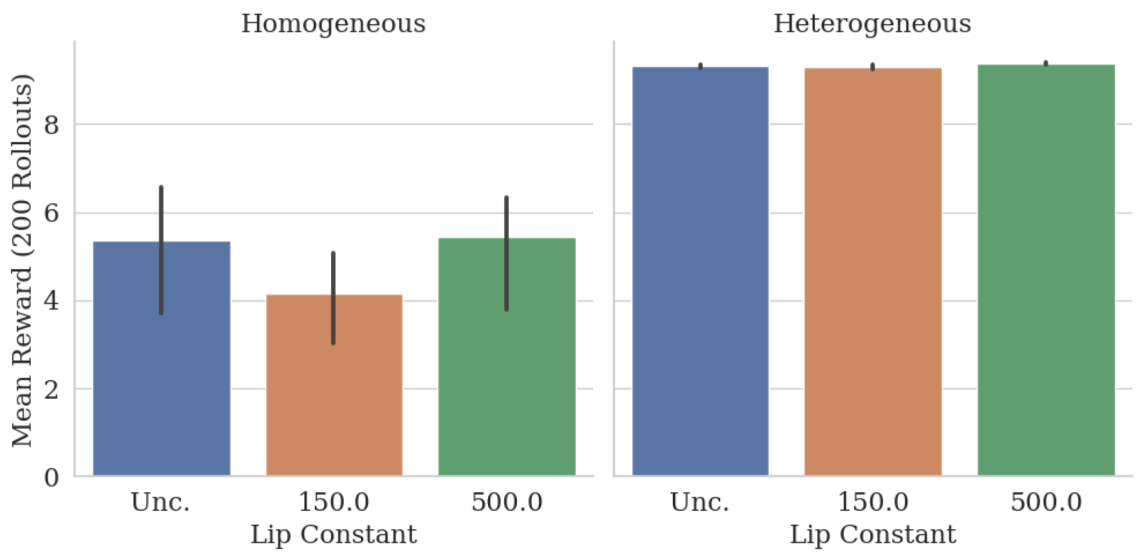


Fig. 4.9 The average reward over 200 rollouts of the three models trained in the Give Way scenario. The error bars show the min and max average value for the three models.

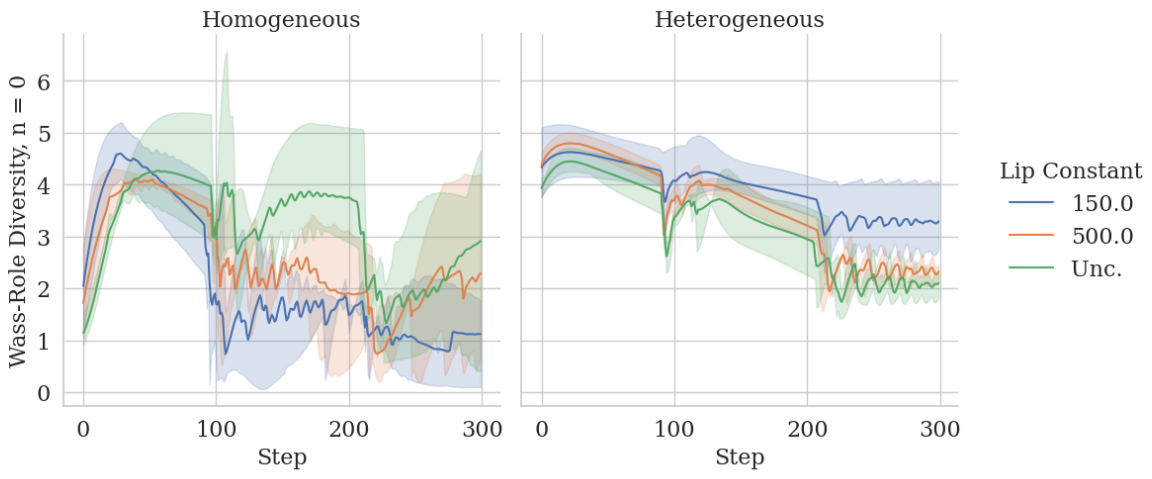


Fig. 4.10 The Wass-Role Diversity for each time step in the Give Way scenario, averaged over 200 rollouts over 300 environments each over 3 seeds.

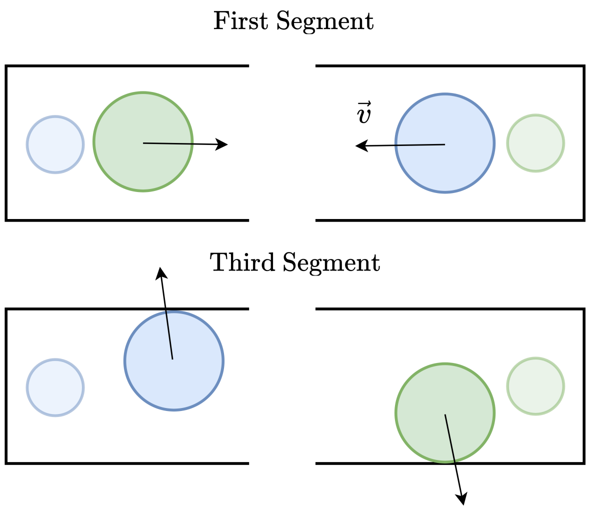
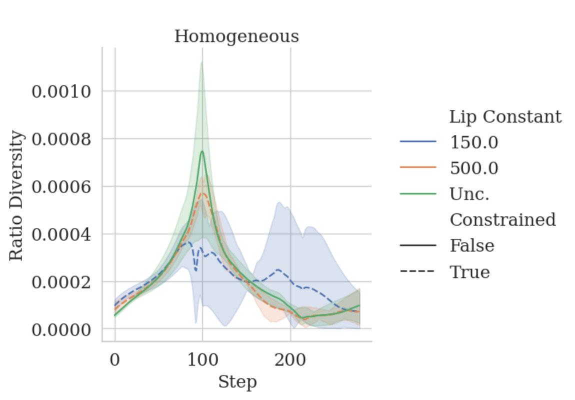


Fig. 4.11 Demonstration of how homogeneous policies can take advantage of different areas of observation space. Only the ends of the corridor are visualized.

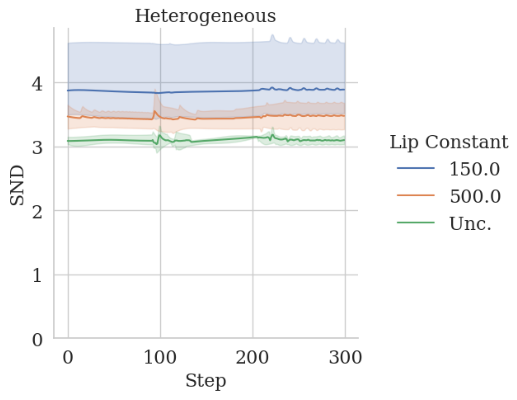
agents continue to increase diversity once again, indicating that they move back towards the other end of the corridor, repeating the cycle shown during the first segment. With $K = 500$, the variance of diversity is very high, indicating that some of the policies learn the optimal policy, while others do not. When $K = 150$, it is clear that the policy breaks down, as the agents do not exhibit a second cycle, and they often get stuck at the center, unable to find a policy where they move across.

Something that is not visible in this plot, but is an interesting phenomenon is that often homogeneous agents under the neural capacity constraints utilized the observation space in creative ways. For example, the heterogeneous policies that successfully trained to get agents to reach the opposite end of the corridor often forced agents to output velocity nearly perpendicular to the wall. This allowed them to use a different area of the observation space to complete their goal, as the velocity of the agent during the first step had been parallel to the corridor. This is shown through a diagram in fig. 4.11.

While it is clear from the plot of the Wass-RD for the heterogeneous policy fig. 4.10 that the diversity increases as the neural constraints are applied, however, from the plot of SND fig. 4.12b, it becomes even more clear as it increases for different neural constraints. As the SND measures the difference between action outputs for the two policies for the same observation, this is a stronger indicator of overall diversity between heterogeneous agents than Wass-RD.



(a) Ratio Diversity d_π, Hom.



(b) SND, Het.

Fig. 4.12 Diversity with different constraint levels in the Give Way scenario for each time step, averaged over 200 rollouts.

For homogeneous agents, the decrease in neural expressivity under constraints is visible through the ratio diversity d_π. It peaks sharply as the agents move near each other without constraints. As the constraint is increased, the spike disappears. The neural constraints limit the maximum ratio diversity of the policy, making it difficult or impossible to learn the optimal, high peak.

4.3.2 Limits on Physical Space and Robustness

In section 3.1.1, we connected Lipschitz continuity and conservative certification. Although the overall Lipschitz constant of these networks is relatively high, we are interested in testing whether constraining the Lipschitz constant of each agent's individually policy translates to an overall improvement in robustness to input perturbations for the system.

To test this theory, we inject uniform observation noise of varying levels. The results are shown in fig. 4.13. The heterogeneous model indicates slightly better robustness to observation noise under the constraints for both $K = 150$ and $K = 500$ due to the increased diversity of the agents and the increased smoothness in their policies. Homogeneous agents with constraints also do not degrade under noise as much as the unconstrained policy. $K = 500$ is the best constraint level for this policy because it does not degrade performance compared to the unconstrained policy and demonstrates robustness to even uniform noise of level 0.5, which is high for the size of the environment.

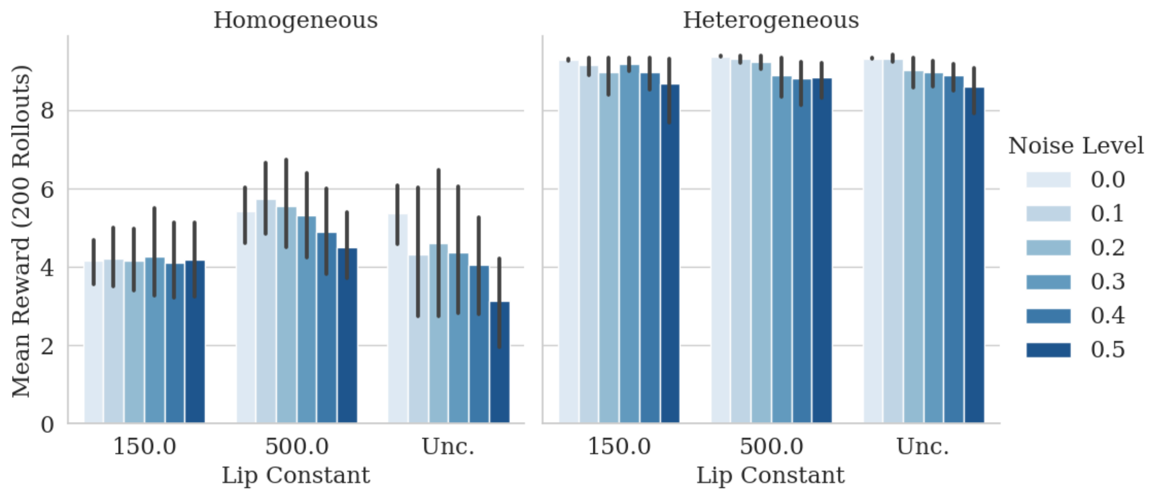


Fig. 4.13 The performance of the Give Way models with different Lipschitz constants levels under uniform noise.

It is likely that using a higher Lipschitz constant for the homogeneous policy in this case would improve performance over the constants shown, but the heterogeneous policy performed much better both in the noise-free case and under uniform noise. Therefore, we can conclude definitive benefits of heterogeneous policies over homogeneous policies in this scenario. This confirms the insights from the Left-Right Scenario that decreasing physical space in the environment is more difficult for homogeneous agents.

In the final section of this chapter, we consider how heterogeneous and homogeneous policies compare when the spatial limits imposed on the agents from the scenario are decreased by meaningfully increasing the feature space.

4.4 Understanding Explicit Indices

4.4.1 Modifying Give Way: Explicit Goal Location

In this section, we investigate whether artificially increasing the richness of the observations of the homogeneous agents can improve agent specialization. Clearly, without an explicit index, the homogeneous agents were not able to specialize into roles in the Give Way scenario as described in section 4.3.

Appending explicit integer IDs to each agent is a common practice that has been used with homogeneous policies to improve diversity (Christianos et al., 2021; Hu et al., 2022). Although this technique increases agent specialization of homogeneous policies, it is not a perfect solution as it has been shown to break down quickly with the number of agents. Further, integer IDs do not make it clear how agents should share their knowledge, as the ID has no real meaning in the scenario. Rather than appending an integer index as many prior works have done, we append the goal position for each agent. This is a much better method for differentiating agents explicitly, as it can change continuously within the environment space. It can benefit learning for all agents as it provides them with knowledge of how to get to a specific goal position that is not dependent on the particular agent that receives this goal location.

4.4.2 Performance Improvements

When the relative goal position between the agent and its goal is appended to the observation, the performance of the homogeneous agents drastically improve, shown in table 4.3. In fact, the homogeneous agents perform better. With the goal position included in the observations, much stronger neural constraints can be used for both the homogeneous and heterogeneous agents than when the goal position is not included. With $K = 1$, both agents have almost no change in performance.

Performance in Modified Give Way			
Policy Type	Lip Constant	Avg Total Reward	Std.
Hom.	1.0	9.539	2.24e-02
	150.0	9.729	3.57e-03
	Unc.	9.727	1.20e-02
Het.	1.0	9.553	2.93e-02
	150.0	9.689	4.30e-02
	Unc.	9.644	5.00e-02

Table 4.3 Average total reward across 200 rollouts for models trained with the goal position appended to the observations in the Give Way scenario.

With the goal position appended to the observations, a number of conditions contribute to homogeneous policies performing better than heterogeneous policies. First, now the homogeneous agents can take full advantage of their shared training examples. Rather than causing instability as the updates from one agent directly opposing the other, the updates from both agents can be used in conjunction to make the policy better. Additionally, both heterogeneous and homogeneous policies improve performance over the models without the goal position provided.

4.4.3 Robustness and Neural Constraints

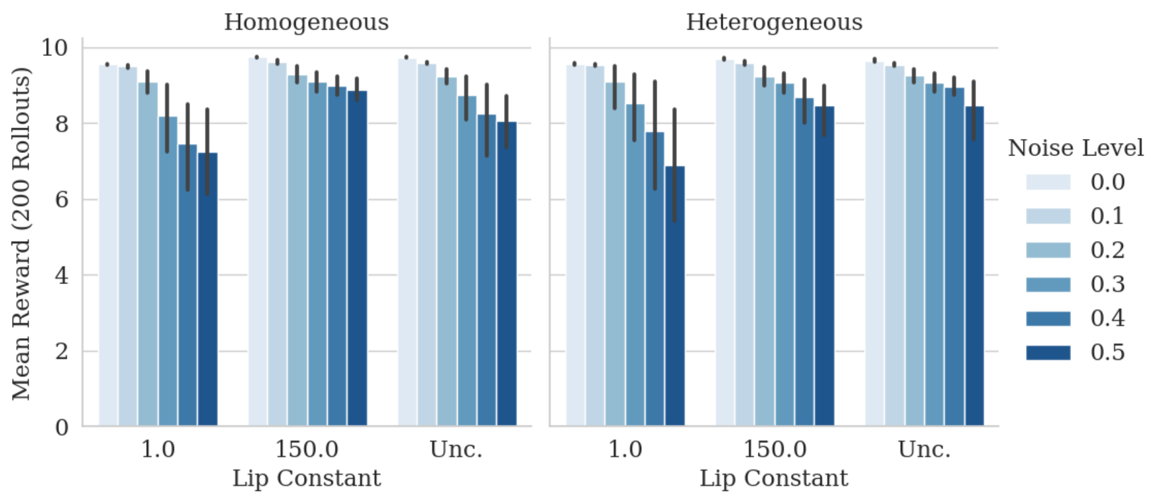


Fig. 4.14 Average total reward across 200 rollouts for models trained with the goal position appended to the observations in the Give Way scenario for different amounts of uniform noise injected into observations between 0 and 0.5.

With the explicit goal position appended to the observation, the neurally constrained agents are now limited in behavior based on distance to the goal position. This limits the benefits of neural constraints. Supporting this, with the explicit goal index appended to the observations, the heterogeneous policy *without* neural constraints now performs the best under uniform observation noise, shown in fig. 4.14.

Additionally, due to the importance of the goal location, it may be expected that the policies of homogeneous agents break down more quickly under injected uniform noise. For the unconstrained models, the heterogeneous model is more robust to noise than the homogeneous model, still showing some benefits of heterogeneity although the homogeneous model performs slightly better in the noise-free case. On the other hand, the constrained

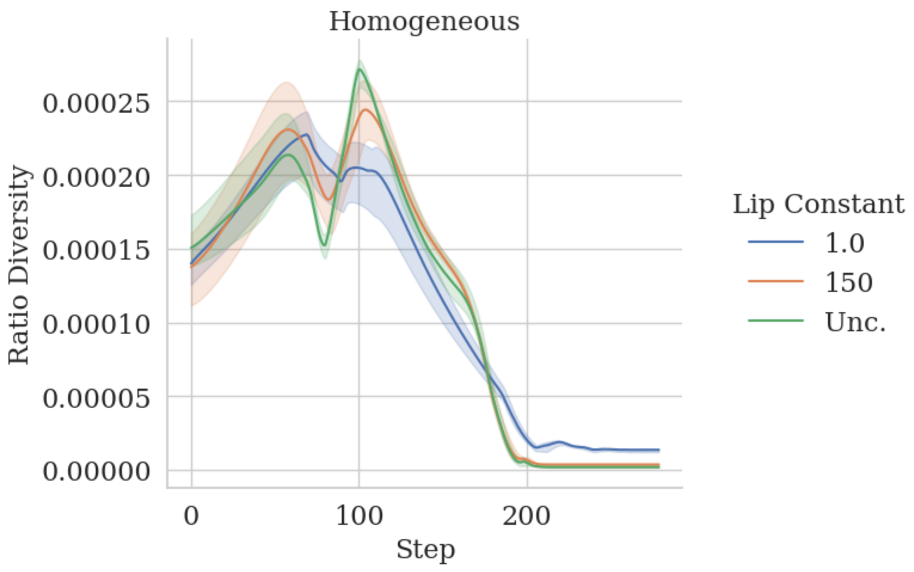


Fig. 4.15 Average d_π for the homogeneous policy in the Give Way scenario with the goal position explicitly given.

homogeneous models perform slightly better under noise than the heterogeneous ones, with the most robust policy being the homogeneous model with constraint $K = 150$.

4.4.4 Neural Constraints and Diversity

Following the insights from injecting observation noise into the policies and testing robustness to input perturbations, we examine the changes in diversity for the different policy types under neural constraints.

The homogeneous policy with the goal location appended can utilize similar neural capacity with the constraints and without, shown in the plot of the average ratio diversity shown in fig. 4.15. In fact, with $K = 150$, the diversity among the agents actually peaks higher than for the unconstrained policy. Given that the level of ratio diversity is still low, this does not indicate a lack of robustness. Overall, the ratio diversity of these policies are ten times smaller than the ratio diversity that was required in fig. 4.12a without the goal position appended.

For both policies, the changes for the Wass-RD between the unconstrained and constrained policies indicate over-smoothing from the $K = 1$ policy, however this is less-so for the homogeneous policy. The SND among agents makes most sense for the optimal policy at $K = 150$ which correlates with the increase in performance.

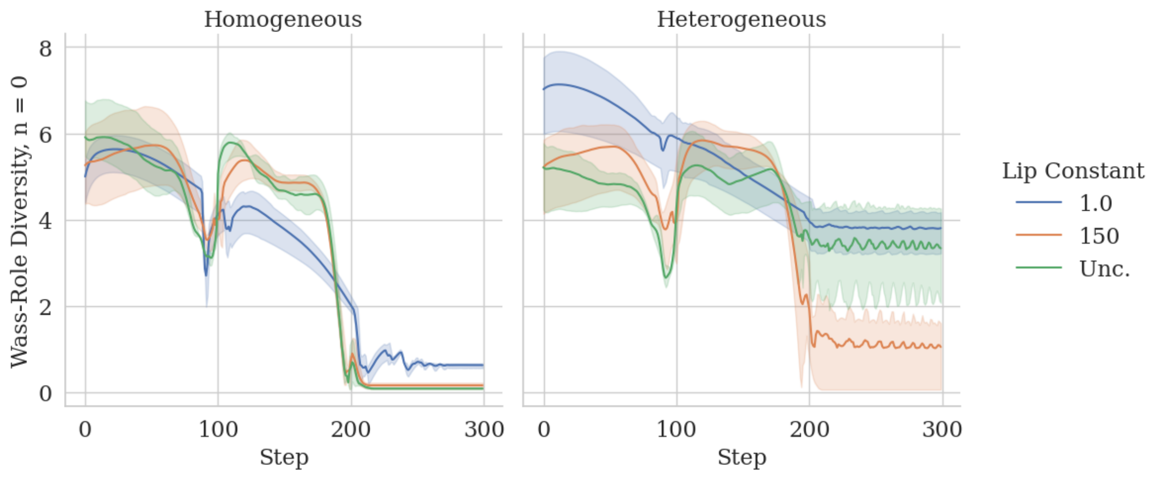


Fig. 4.16 Average Wass-RD across 200 rollouts with 300 environments each over 300 steps per environment, for models trained with the goal position appended to the observations.

Adding features to the observations which are continuous in the environment and aid agents in specialization can improve the ability of homogeneous policies to specialize.

4.5 Concluding Insights

In this chapter, we demonstrated how imposing neural constraints was beneficial to learning cooperative multi-agent policies with agent specialization. The the Lipschitz constant of the policy networks were properly tuned to the given the scenario, the neural constraints were showed to increase diversity.

Despite the fact that homogeneous policies can learn the optimal policy in many cases, there are specific degenerate cases where homogeneous policies cannot learn the optimal policy. Increasing the feature space of the agents can be an useful technique to model multiple behaviors. However, adding features to the observation space of the agents may not be suitable for every environment because the features should be continuous and relevant to the goals of the agents in order to support training.

When the environment configuration places limits on the agents in terms of physical space or observation noise, heterogeneous agents should be heavily considered to model multiple roles, as heterogeneous agents are not limited by functional constraints. Further, heterogeneous agents under cognitive limitations improved in robustness, optimization and performance.

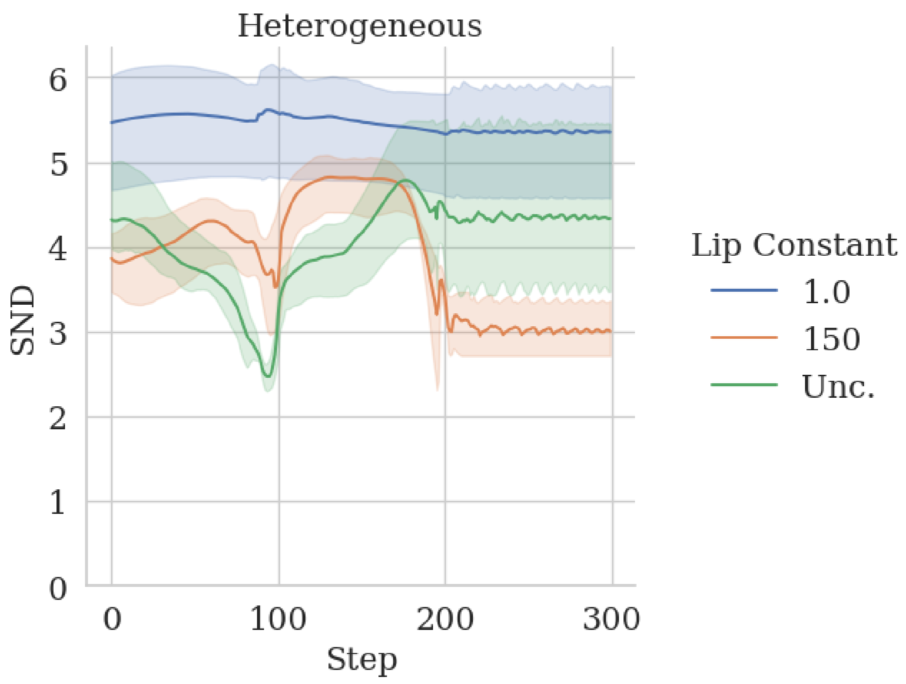


Fig. 4.17 Average SND across 200 rollouts with 300 environments each over 300 steps per environment, for models trained with the goal position appended to the observations.

Chapter 5

Behavioral Diversity with Physical Differences

In this chapter, we explore how physical difference between agents can be utilized when learning diverse behavior.

5.1 Introducing the Two-Mass Scenario

Building off of the insights from the previous chapter, the Two-Mass scenario is designed to test the learning capabilities of heterogeneous and homogeneous policies when the agents have different physical types. The two agents are spawned randomly from a uniform distribution within a square region with side length d. The first agent has mass in the range $[1.5-2.5]$ and the second agent's mass is in the range $[3.5-4.5]$, chosen randomly at uniform. The goal of the scenario is to maximize the speed of a single agent while minimizing energy cost.

$$R = \max_{i \in \mathcal{N}} \{v_i\} - 0.17 \cdot \frac{1}{\sqrt{2}} \sum_{i \in \mathcal{N}} \|u\| \tag{5.1}$$

In the Two-Mass scenario, the agents are rewarded for maximizing the speed of a single agent while minimizing their overall energy expenditure. This means that the heavier agent should not move at all, while the lighter agent should output maximum force. For simplicity, the agents are limited to only move horizontally $\mathbf{u} = u_x$. To solve this scenario optimally, the

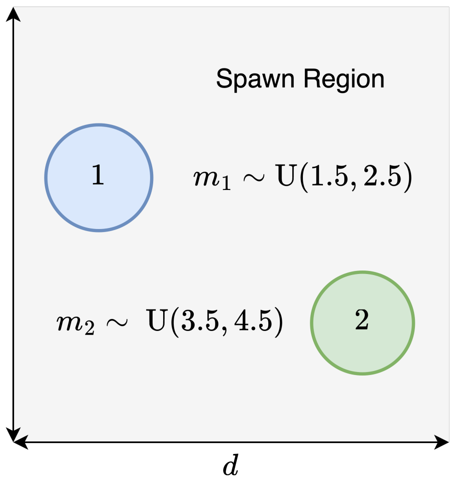

Spawn Region
1
$m_1 \sim \mathrm{U}(1.5, 2.5)$
$m_2 \sim \mathrm{U}(3.5, 4.5)$
2
d

Fig. 5.1 A diagram describing the Two-Mass scenario environment.

agents must specialize into either the role of the lighter agent (output a non-zero force) or the role of the heavier agent (output no force).

The optimal policy requires strictly unique behavior from each agent. For agents to behave optimally, the required policy is very simple. In fact, the optimal force output for each agent is constant:

$$u_x^{(\text{Heavy})} = 0, \;\; u_x^{(\text{Light})} = u^{\text{ opt}}$$

While the optimal policy per agent in this scenario is extremely simple, only heterogeneous agents can take advantage of this simplicity by learning only one behavior per agent. Through training, the heterogeneous policy can learn the best policy for the corresponding physical type because the physical difference is clear based on the observations of each agent and the difference in reward.

Homogeneous agents cannot take advantage of this simplicity and must learn which observations under the given policy are only observed by the light agent $\mathcal{O}_{(\text{Light})}$ and which observations are only observed by the heavy agent $\mathcal{O}_{(\text{Heavy})}$ so that $\pi(o_1) \approx 0$ for $o_1 \in \mathcal{O}_{(\text{Light})}$ and $\pi(o_2) \approx u^{\text{ opt}}$ for $o_2 \in \mathcal{O}_{(\text{Heavy})}$. Between these regions, the policy must learn actions which will cause the agents to behave differently according to their physical type to separate the observations that they might see during a rollout. Then the difference in observation distribution for each agent can be used to specialize. Homogeneous policies must be much more complex, and must have high neural expressivity.

Two-Mass Scenario Configuration	
Reward Type	Shared
Max Velocity Reward	$\max_{i \in \mathcal{N}}\{v_i\}$
Energy Penalty	$-\frac{1}{\sqrt{2}} \sum_{i \in \mathcal{N}} \lVert u \rVert$
Energy Penalty Coefficient	0.17
Collisions Possible	No
Num. Agents	2
Agent Mass	$m_1 \sim \mathrm{U}(1.5, 2.5)$
Agent Mass	$m_2 \sim \mathrm{U}(3.5, 4.5)$
Agent Radius	0.15
Agent Actions	$a_i^t = [u_x^t, 0]$
Spawn Region	$d = 1$

Table 5.1 The configuration of the Two-Mass scenario. The agents are the same physical type. The agents get reward proportional to their distance from goal at each time step. When both agents are on their respective goal, they get a final reward R_g. The goal per agent is set before training.

5.1.1 Training Two-Mass

This high neural expressivity requirement is shown in fig. 5.2. As the Lipschitz constant is decreased, the homogeneous policy breaks down, while the heterogeneous policy improves under a lower Lipschitz constant. Heterogeneous polices increase performance under these constraints as the policy is forced to output more similar actions for each observation for each agent. Homogeneous polices decrease performance under all constraints and increasing the strength of the constraint is correlated with a decrease in performance.

Figure 5.3 shows the difference in policy plots between the homogeneous and heterogeneous agents. In fig. 5.3a, around the line $v^{(\mathrm{Light})} = v^{(\mathrm{Heavy})}$, the policy gives both agents a small, equal amount of force. Once the speed of a single agent increases more than another, the output force quickly increases for only the agent with higher speed. This corresponds to the optimal homogeneous policy. The policy must output the same action for the same observation per agent, so it is forced to be symmetric for all agents. With a strong constraint

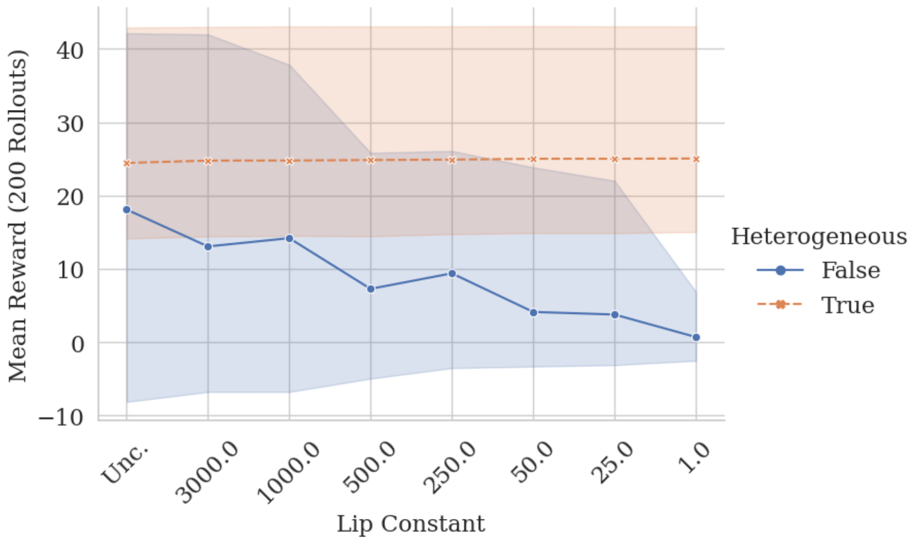
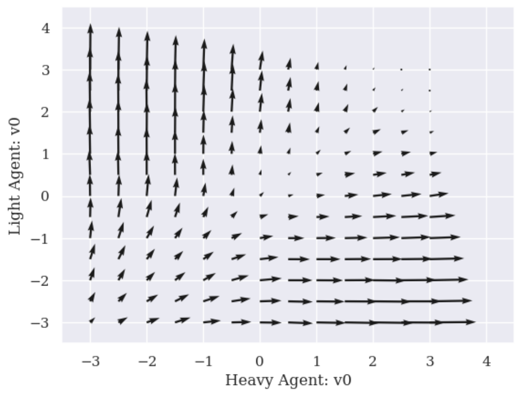


Fig. 5.2 Mean total reward in the Two-Mass scenario of both policy types under different neural constraints. Three different models are trained for each lip constant for 500 iterations. The mean total reward is averaged over 200 rollouts over 200 environments per rollout per model. The error bars show the minimum average and maximum average across the three models.

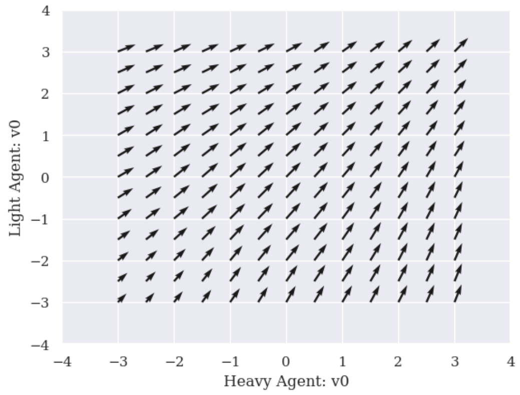
of $K = 1$ shown in fig. 5.3b, the policy is overly smoothed and outputs similar force to both agents even when their velocities are different. In contrast, fig. 5.3c shows that the heterogeneous policy simply outputs a non-zero constant force for the light agent and no force onto the heavy agent, a much simpler policy. Given that it is constant, this policy is unchanged given a strong Lipschitz constant of $K = 1$.

Both homogeneous and heterogeneous policies are able to learn these agent specializations. However, Bettini et al. (2023a) noticed that although the policy could learn heterogeneous behavior, the learned policy was not robust to observation noise. The single homogeneous policy encodes both specialized behaviors, so under observation noise the agents can receive observations that make them act according to the wrong behavior. This is confirmed by the fact that the homogeneous policies break down under a relatively high Lipschitz constant.

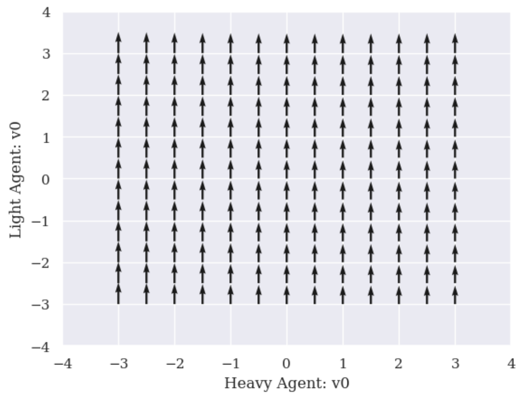
The training plots of both heterogeneous and homogeneous policies under different levels of neural constraints are shown in fig. 5.4. For all Lipschitz constants imposed, heterogeneous agents learn the optimal policy within 100 iterations, but averages around 40 iterations. Without neural constraints, the homogeneous policy converges in around 300 iterations. In the unconstrained case, both the heterogeneous and homogeneous policies learn the optimal policy. With the neural constraints imposed, the homogeneous policies take longer



(a) Hom. $K = 3000$



(b) Hom. $K = 1$



(c) Het. $K = 1$

Fig. 5.3 Plots of the learned homogeneous policy at $(0,0)$ depending on the speed of the agents. The magnitude of the arrow along each axis represents the policy output force for a single agent.

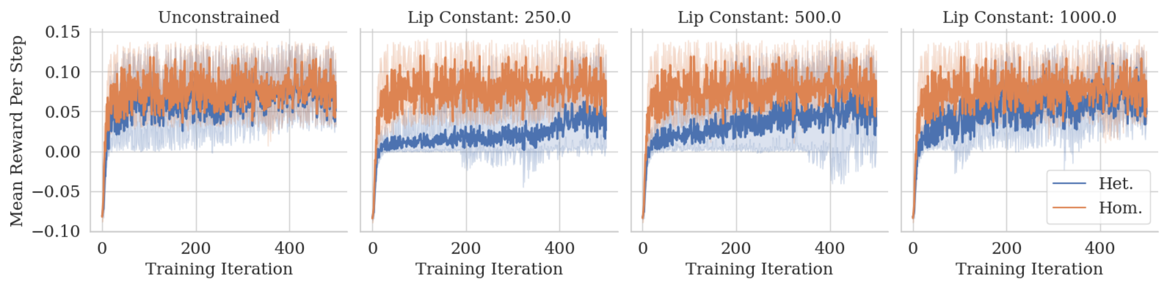


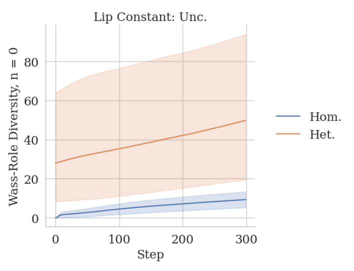
Fig. 5.4 Training plots of the Two-Mass scenario for selected Lipschitz constants. Each training plot shows the average reward per step over 100 steps, for each training iteration, averaged over three models with different seeds.

time to train for stronger constraints. With constraints $K \leq 500$, the optimal homogeneous policy is not learned within 500 iterations. However, decreasing the neural constraints on the heterogeneous policy slightly decreases training time. This increase in training time for homogeneous policies is an additional indicator that the neural constraints cause the homogeneous policies to break down.

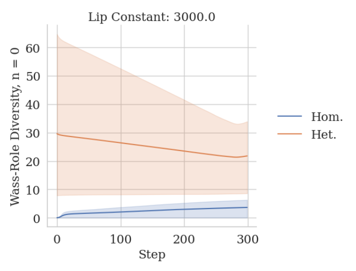
5.1.2 Evaluating Two-Mass Diversity

Figure 5.4 shows that homogeneous policies take a very long time to converge under any constraint, even very high ones. The policy is forced to output more similar actions for similar observations, which decreases the proximity of the two separate observation regions $\mathcal{O}_{(\text{Light})}$ and $\mathcal{O}_{(\text{Heavy})}$. Figure 5.6 shows the average difference in observations between the agents throughout the steps of a rollout. For homogeneous policies, higher neural constraints decrease the rate at which the difference in observations between agents changes. Heterogeneous agents separate at nearly the same rate for all constraints. Homogeneous unconstrained policies can learn to separate the agents observations, but the neural constraints decrease observation difference by the end of the rollout. In homogeneous policies, observation difference is a proxy for diversity as the homogeneous policy will always evaluate to be the same for all inputs. The reduction in observation difference also indicates a reduction in diversity. The heterogeneous policy, on the other hand, has the same observation difference for all neural constraint levels.

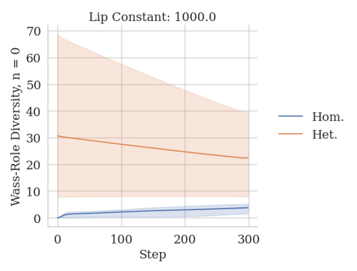
Additionally, fig. 5.5 shows that the diversity of the agents increases as the scenario progresses in the homogeneous scenarios, but is much less diverse than in the heterogeneous policies. Homogeneous diversity ratings are lower overall than heterogeneous policies, even without



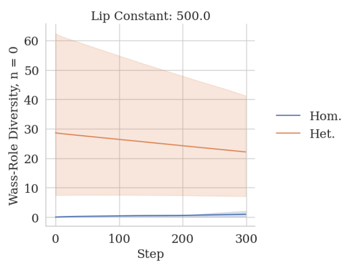
(a) Unconstrained.



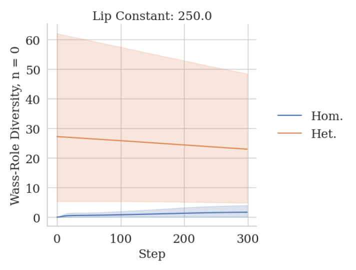
(b) $K = 3000$.



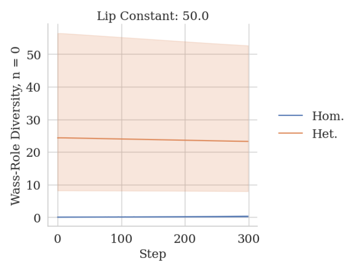
(c) $K = 1000$.



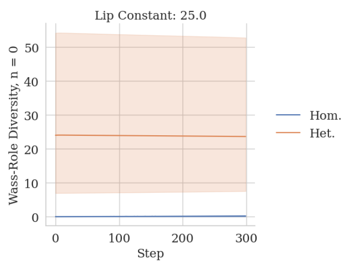
(d) $K = 500$.



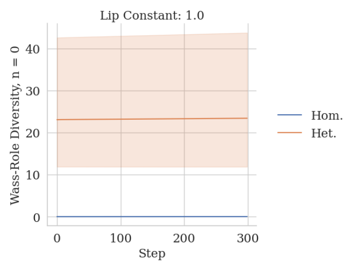
(e) $K = 250$.



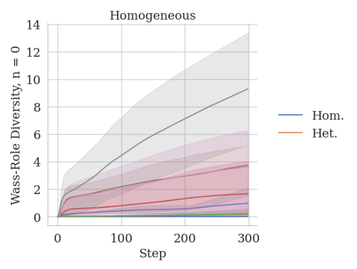
(f) $K = 50$



(g) $K = 25$



(h) $K = 1$



(i) Homogeneous policies.

Fig. 5.5 The Wasserstein-Role Diversity metric with $n = 0$ for all neural constraint levels for the Two-Mass scenario evaluated for each step of a rollout averaged over 200 rollouts across three models

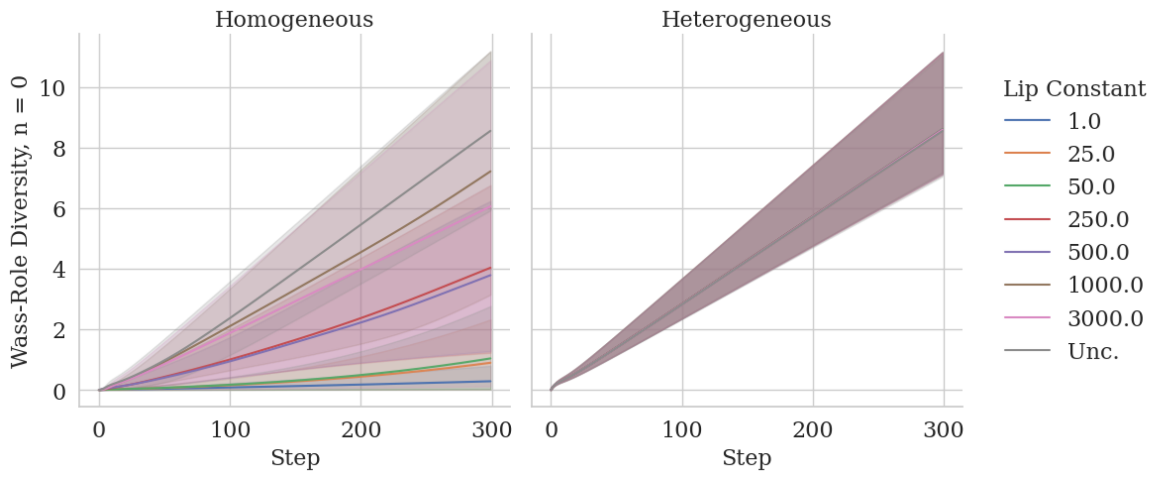


Fig. 5.6 Observation distance in the Two-Mass policies for each step of a rollout averaged over 200 rollouts across three models.

constraints. Figure 5.5i shows that the maximum Wass-RD reached by the homogeneous agents is between 5 and 15, while most heterogeneous policies reached between 8 and over 50. The constraints imposed on the heterogeneous policies makes the diversity between agents more consistent throughout the rollout. This corresponds to a more optimal policy, as the output policy should be constant for all inputs. It also decreases the variance in diversity. For the strongest constraint of $K = 1$ (fig. 5.5h), the minimum diversity rating for the heterogeneous policy is the highest. In contrast, the homogeneous policy has average diversity near zero throughout all time steps.

In addition to the raw difference in diversity between the heterogeneous and homogeneous agents, the homogeneous diversity ratings evolve differently throughout the rollout. The diversity among homogeneous agents start at zero at the start of each rollout when the types of each agent are unknown. Increasing the constraints on the homogeneous policy decreases diversity, and the diversity score decreases as the Lipschitz constant is decreased.

Figure 5.7 shows that without constraints, the ratio of diversity is very high as the agents are meant to act in two specialized ways with similar observations.

5.1.3 Robustness to Noise

The link between the neural expressivity of the policy and the Lipschitz constant of the network was presented in section 3.1.2. Evaluating the model performance for different

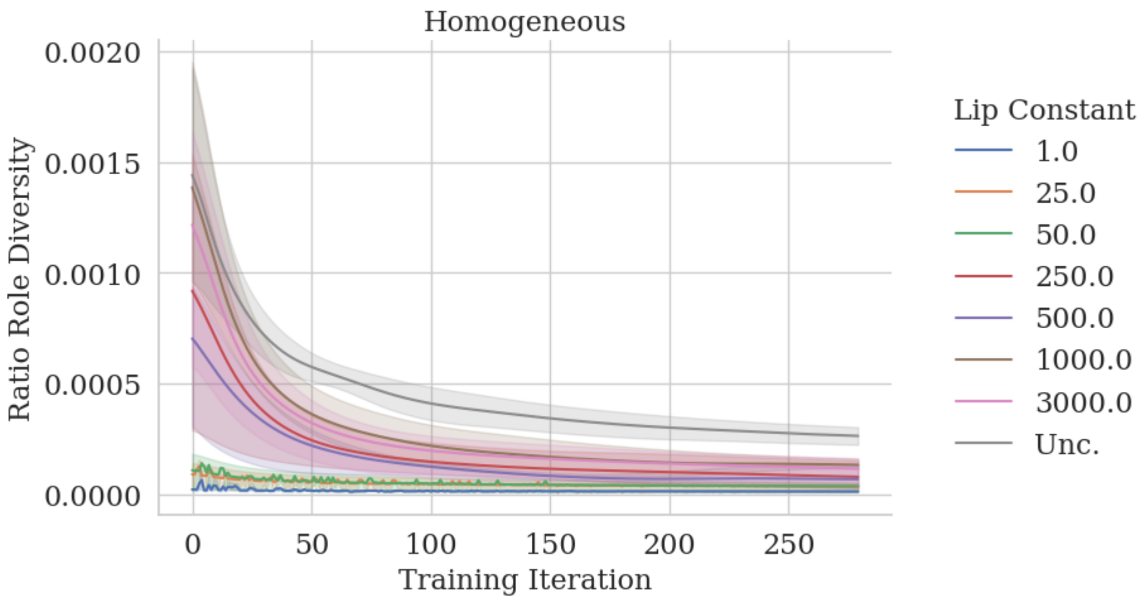


Fig. 5.7 Ratio diversity d_π for the homogeneous policy evaluated for each step of a rollout averaged over 200 rollouts across three models.

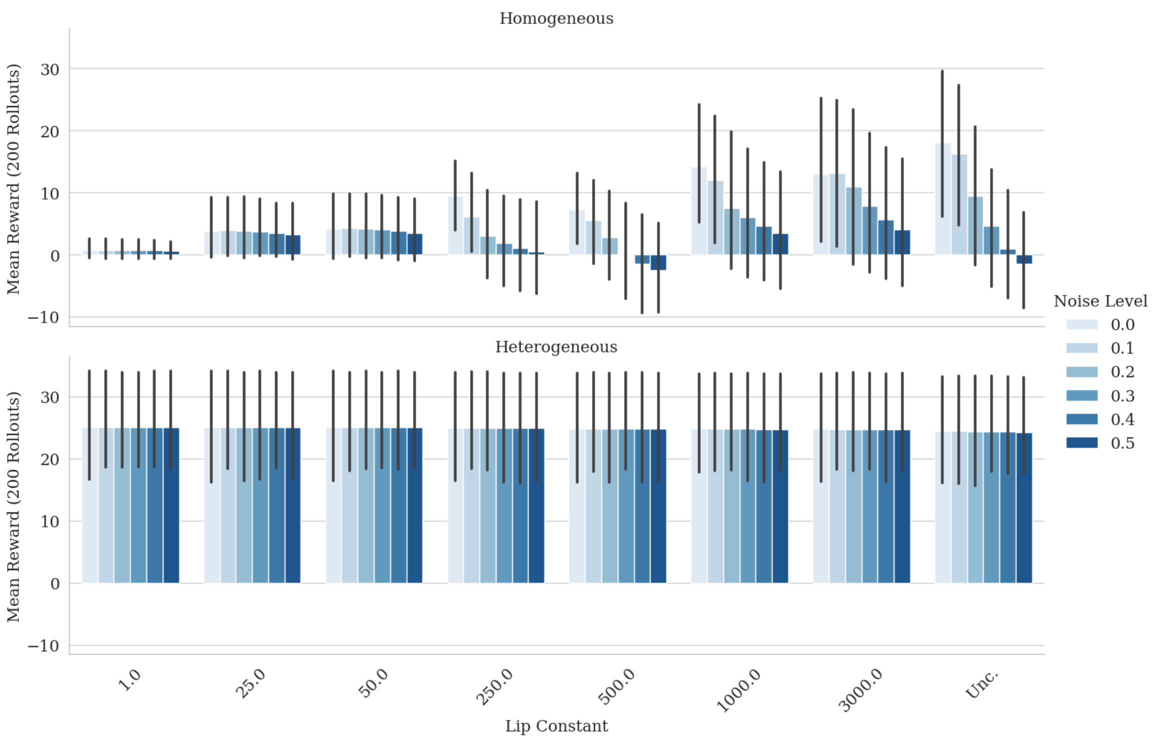


Fig. 5.8 Mean total reward over 200 environments after 300 steps averaged over 3 models trained with different seeds in the Two-Mass scenario.

neural constraints under observation noise demonstrates this link. The policies which are able to learn under stronger neural constraints are better able to handle higher levels of observation noise. Through Figure 5.8, it is clear that the heterogeneous policy is nearly invariant to observation noise. For $K = 1$, the policy shows almost now change for uniform noise up to 0.5. This is due to the fact that the policy is smoothed and represents the constant action output accurately. In contrast, the homogeneous policy breaks down almost immediately under observation noise, even in the unconstrained case where the performance in a noise-free setting nearly matches the heterogeneous model.

5.1.4 Varying Mass Difference and Diversity

As the difference in mass increases between the agents, the required neural expressivity of the homogeneous policy decreases. To explicitly test the ability of the homogeneous policy to use the physical difference between agents to specialize, the spawn region between the agents is set to $d = 0$ and both agents are spawned at the same location. Additionally, the mass noise of each agent is reduced to 0, so the agents are spawned with the same mass each time.

Through these different mass differences, we can gain an understanding of how different conditions affect the level of specialization that is achievable under the agent physical types and neural constraints.

Modifications to the Two-Mass Scenario	
Num Agents	2
Agent Mass m_1	2.0
Agent Mass m_2	2.0, 6.0 or 10.0
Spawn Region	$d = 0$

Table 5.2 The modifications to the Two-Mass scenario configuration. In this modified scenario the agents are spawned at the same location. Multiple experiments are run which vary the second agent's mass (set before training).

Through the Two-Mass scenario, the impacts of neural constraints on learning, performance, and diversity are introduced. Next, the scenario features within Two-Mass that contribute to

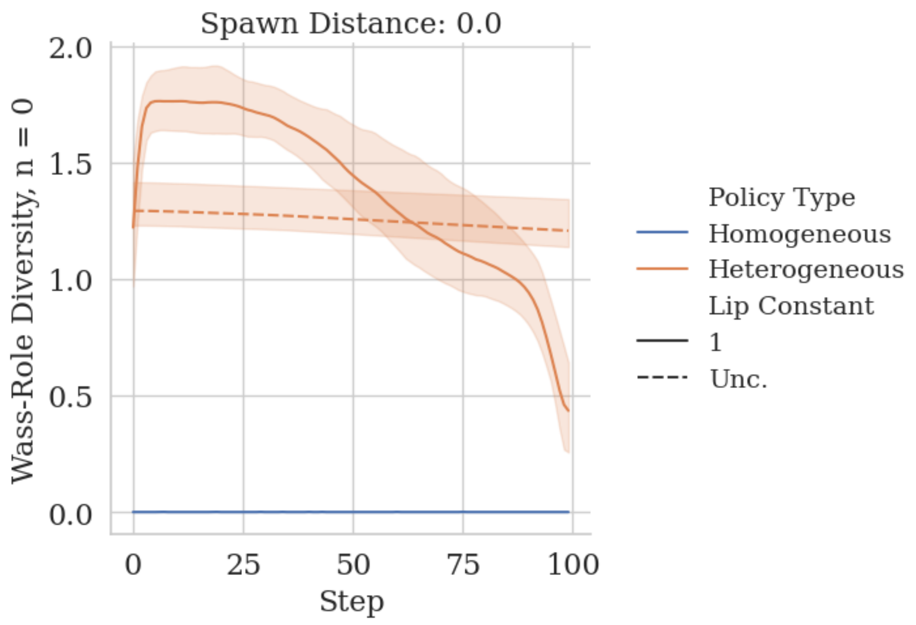


Fig. 5.9 The Wass-Role Diversity of agents through the Two-Mass scenario with no mass difference between agents averaged over 200 rollouts with 300 environments each over 3 different model seeds.

the difficulty of learning with a homogeneous policy in this scenario are broken down. First, the parameters of this scenario are varied to change the mass difference between the agents.

Zero Mass Difference

When both agents have the same mass and they are spawned at the same location, the agents cannot reliably separate under a homogeneous policy. This is the same principle addressed in the Left-Right scenario with no spawn separation (section 4.2.3).

Throughout the entire rollout, the average action diversity between agents using a homogeneous policy is zero, shown in fig. 5.9. On the other hand, the heterogeneous agents are able to learn multiple roles under these conditions. This result seems trivial, but it clarifies the ways in which homogeneous policies are *inadequate* for learning multiple roles.

While heterogeneous agents are able to solve the scenario with an unconstrained policy, the unconstrained policy learned is noticeably less smooth. Under these scenario conditions, there are actually two optimal policies. There is no right answer as to which agent should be the one to stay stationary and which should have non-zero velocity. When one agent learns

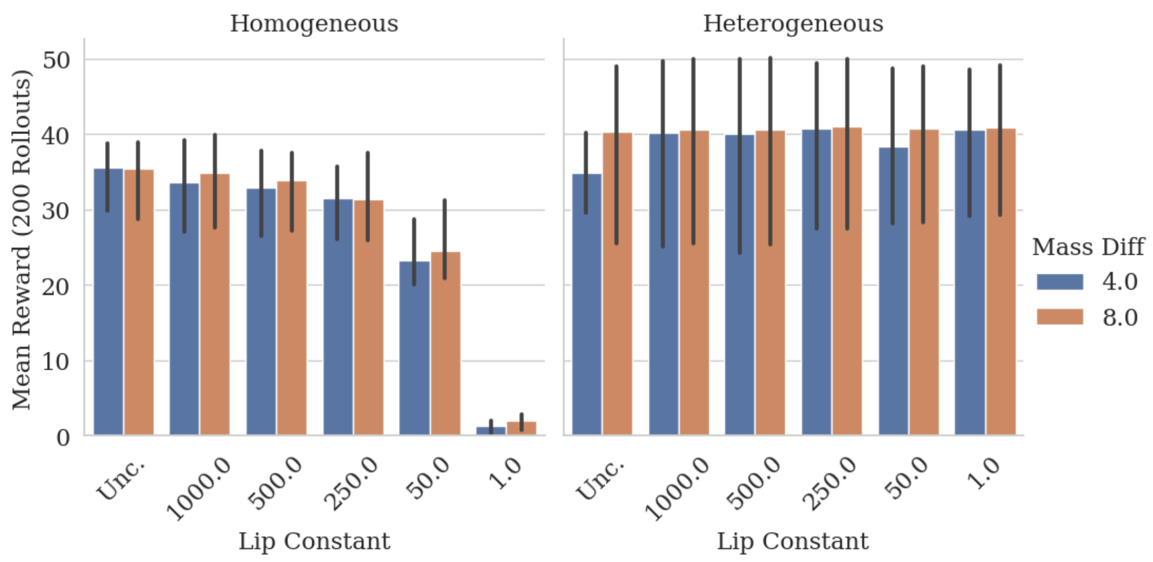


Fig. 5.10 Mean reward over 200 rollouts across 300 steps per rollout. Rollouts are conducted for the three trained models per Lipschitz constant and mass difference pairing. The error bars show the minimum and maximum of these rollouts.

to increase its velocity such that it moves faster than the other, the optimization direction for both agents suddenly shifts, as the other agent should then learn to have zero velocity. This can cause the policy to be non-stationary during training. When both agents have similar velocity, it is difficult for the optimization objective to fall into one of the global optimums. In this case, the neural constraints on heterogeneous agents provide optimization benefits, as it is harder for each agent to dramatically change its learned actions during training. Figure 5.9 shows that the diversity of the heterogeneous policy under the constraint is almost constant, while the unconstrained policy after the same number of iterations varies greatly throughout the rollout. As with the original Two-Mass scenario parameters, the optimal policy corresponds to a constant diversity between agents throughout the rollout.

Varying Non-Zero Mass Difference

In the previous section, we decreased the physical differences between agents. Now, we consider increasing the mass difference between agents. We expect to see that as the physical differences are increased, learning the optimal policy becomes easier. If the physical difference between agents is higher, the same force input will produce a larger difference in observations between the agents.

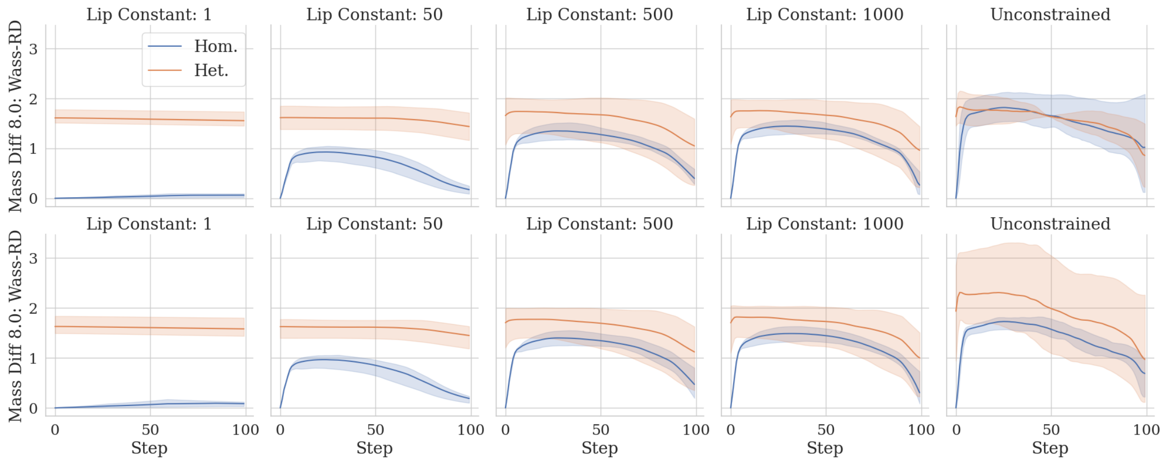


Fig. 5.11 The Average Wasserstein Role Diversity ($n = 0$) throughout a rollout within the Two-Mass scenario with mass differences of 4.0 and 8.0, averaged over 200 rollouts over 300 environments.

Figure 5.10 shows that the performance of the homogeneous policy with mass difference 4.0 slightly decreases for all constraints compared to the policies with mass difference 8.0. This indicates that the policy with lower mass difference requires slightly more policy expressivity.

Within the heterogeneous policies, although most policies learn the optimal policy, the unconstrained model with lower mass difference has lower performance than the model with higher mass difference. This is because as the mass difference increases, the optimization problem becomes easier, as was noticed when optimizing the policy with no mass difference.

Despite the slight increase in performance of the policies under the mass difference of 8.0 over the mass difference of 4.0, the diversity of the policies are relatively the same under constraints. The biggest difference in diversity between the two policies is for the unconstrained models. The benefits of heterogeneity under constraints are still visible under these relaxed mass differences.

5.2 Concluding Insights

In the Two-Mass scenario, there is a clear need for diverse behavior. Representing this diverse behavior with a homogeneous policy is possible, but the level of diversity that can be achieved is limited by the fact that the policy must utilize actions to determine which role a given agent should take. Further, learning multiple behaviors within a single homogeneous

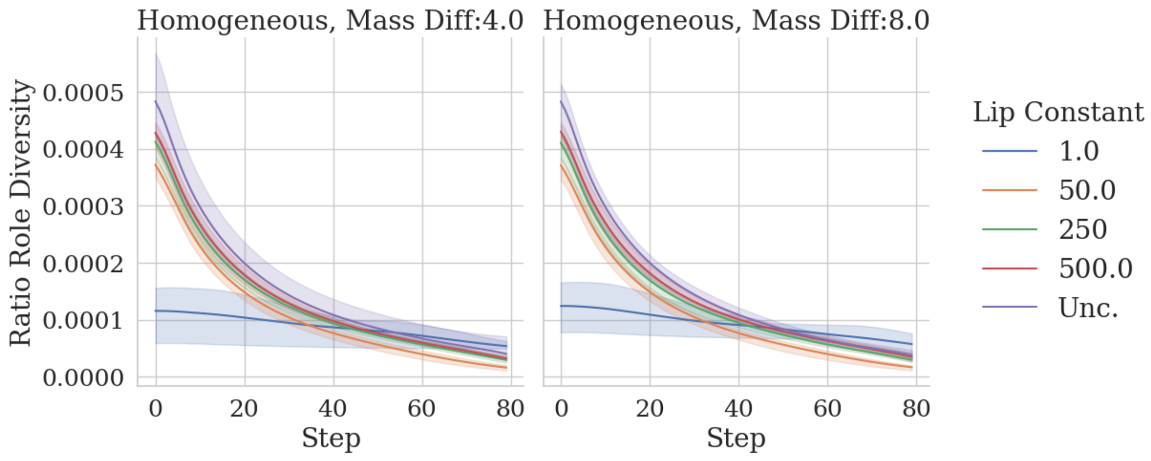


Fig. 5.12 The ratio diversity of the policies learned with mass difference 4.0 and 8.0.

policy made the policy less robust, as shown by the increase in d_π and lack of robustness to noise. By understanding how the homogeneous policy breaks down in this scenario, it is more clear how the neural capacity constraints impact the behaviors and roles of the agents. For nearby observations, agents must output similar actions. These neural constraints smooth the policy function and regularize the *behavior* of the agents. In heterogeneous policies, this can be beneficial, as was seen in the stabilization of diversity between agents throughout all time steps.

Crucially, the homogeneous agents are not able to effectively encode multiple roles within a single, constrained policy. This is an important result which indicates that this constraint method is effective at enforcing diversity. Further, the strength of the neural constraint correlates with the decrease in diversity in the homogeneous policy, meaning that the the level of diversity allowed in a single policy can be tuned. Using this method, the effective neural capacity of each agent can be reduced to a level suitable for the environment.

This scenario is extremely specialized and simple, as it only requires learning a constant policy using heterogeneous agents. Therefore, scenarios that require more complex actions must be considered. In this scenario, the diversity of the heterogeneous agents in the unconstrained case in this scenario was already near optimal. Even still, although this method did not strictly increase diversity between agents, it caused the diversity between agents to stabilize and the variance to decrease. Under very strong constraints, the minimum diversity increased. Secondarily, this method is was shown to slightly increase robustness of the heterogeneous policies and slightly improve optimization, even in a case where the optimal policy was extremely simple.

Chapter 6

Conclusion

6.1 Discussion

This work contributes to a broader goal within multi-agent cooperative reinforcement learning to understand how to model collaboration in complex systems. The conclusions of this work are twofold.

First, limiting the Lipschitz constant of the policies of heterogeneous agents is shown to increase diversity. By bounding the maximum rate of change of each agent's policy, it forces each agent to learn simpler, smoother behavior. While both homogeneous and heterogeneous policies can benefit from smoother policies, heterogeneous policies in particular can improve under strong neural constraints. By forcing simpler policies, neural constraints naturally encourage the agents to take on unique roles in a scenario. Further, teams of agents with limited neural expressivity improve in both optimization and robustness when the constraints are chosen at an optimal level.

Second, while many previous works have utilized homogeneous policies in settings that require multiple roles, these types of policies have specific limitations that have not yet been properly understood. This work outlines these limitations. Homogeneous policies are limited by function constraints: for every observation, only a single action distribution can be learned. While obvious, this makes learning multiple roles *impossible* in scenarios where agents share observation space under a homogeneous policy. When homogeneous policies *can* learn multiple role specializations, heterogeneous policies can often do so with lower neural expressivity, taking advantage of the benefits described above.

Heterogeneous policies are not suitable for every situation and have some limitations. Scenarios which require high degrees of exploration or agents can act independently without sharing the same region of observation space in the optimal policy, parameter sharing provided by homogeneous policies can improve performance and increase data efficiency. However, this work presents specific scenario conditions under which heterogeneous policies provide a large improvement over homogeneous policies.

6.2 Future Work

There are many avenues of research that can build upon the benefits of heterogeneity and policy expressivity constraints presented in this work. While this work found definitive results for the scenarios presented, the scenarios chosen are intentionally simple. Only scenarios with two agents and two unique specializations were considered. In the future, it would be interesting to explore how neural constraints may affect more complex scenarios or tasks. For more complex situations, it may be useful to consider how agents with varying levels of neural capacity can work together in a system.

Additionally, there are many alternative options that can be used to constrain the Lipschitz constant of a neural network. For example, different constants can be applied to each layer, or the norm type can be varied.

Large teams of heterogeneous agents may be infeasible when each has its own set of parameters. However, decreasing each agents' neural capacity may be able to help scale heterogeneous policies across larger teams of agents. Learning Lipschitz continuous polices should help this goal given the benefits in optimization and increased diversity.

Finally, previous works have shown how heterogeneous agents can reduce the Sim-to-Real gap when transferring simulated results onto real robots (Bettini et al., 2023a). Future experiments should consider how decreasing the neural capacity of heterogeneous agents can reduce the additional training time required to transfer a policy to real robots.

References

Cem Anil, James Lucas, and Roger Baker Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:53292059.

Tucker Balch. Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots*, 8(3):209–238, Jun 2000. ISSN 1573-7527. doi: 10.1023/A:1008973424594.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning, 2022.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 1485–1494, Richland, SC, 2023a. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. System neural diversity: Measuring behavioral heterogeneity in multi-agent learning, 2023b.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. *CoRR*, abs/2102.07475, 2021. URL https://arxiv.org/abs/2102.07475.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cohen19c.html.

Christian Schröder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *CoRR*, abs/2003.06709, 2020. URL https://arxiv.org/abs/2003.06709.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Buşoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, 2021a. URL https://api.semanticscholar.org/CorpusID:234357797.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Buşoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, 2021b. URL https://api.semanticscholar.org/CorpusID:234357797.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, Feb 2021. ISSN 1573-0565. doi: 10.1007/s10994-020-05929-w.

Niko Grupen, Natasha Jaques, Been Kim, and Shayegan Omidshafiei. Concept-based understanding of emergent multi-agent behavior. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=zt5JpGQ8WhH.

Siyi Hu, Chuanlong Xie, Xiaodan Liang, and Xiaojun Chang. Policy diagnosis via measuring role diversity in cooperative multi-agent RL. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9041–9071. PMLR, 2022. URL https://proceedings.mlr.press/v162/hu22c.html.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/b710915795b9e9c02cf10d6d2bdb688c-Abstract.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, May 1998. ISSN 00043702. doi: 10.1016/S0004-3702(98)00023-X.

Ouail Kitouni, Niklas Nolte, and Mike Williams. Robust and provably monotonic networks. *ArXiv*, abs/2112.00038, 2021. URL https://api.semanticscholar.org/CorpusID:244772981.

Taisuke Kobayashi. L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4032–4039, 2022. URL https://api.semanticscholar.org/CorpusID:246863906.

Chenghao Li, Chengjie Wu, Tonghan Wang, Jun Yang, Qianchuan Zhao, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235352611.

Xiaobai Ma, Jayesh K. Gupta, and Mykel J. Kochenderfer. *Normalizing Flow Policies for Multi-agent Systems*, volume 12513 of *Lecture Notes in Computer Science*, page 277–296. Springer International Publishing, Cham, 2020. ISBN 978-3-030-64792-6. doi: 10.1007/978-3-030-64793-3_15. URL https://link.springer.com/10.1007/978-3-030-64793-3_15.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. URL http://arxiv.org/abs/1802.05957.

Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 1810–1816, May 2021. doi: 10.1109/ICRA48506.2021.9561138.

Mícheál Ó Searcóid. *Metric spaces*. Springer Undergraduate Mathematics Series. Springer, Guildford, England, 2007 edition, Sept 2006.

Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7512–7519, 2021. doi: 10.1109/IROS51168.2021.9635857.

David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. *CoRR*, abs/1610.01945, 2016. URL http://arxiv.org/abs/1610.01945.

Amanda Prorok, M. Ani Hsieh, and Vijay Kumar. Formalizing the impact of diversity on performance in a heterogeneous swarm of robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, page 5364–5371, May 2016. doi: 10.1109/ICRA.2016.7487748.

Amanda Prorok, Matthew Malencia, Luca Carlone, Gaurav S. Sukhatme, Brian M. Sadler, and Vijay R. Kumar. Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems. *ArXiv*, abs/2109.12343, 2021. URL https://api.semanticscholar.org/CorpusID:237941071.

Craig W. Reynolds. *Flocks, Herds, and Schools: A Distributed Behavioral Model*, page 273–282. Association for Computing Machinery, New York, NY, USA, 1998. ISBN 158113052X. URL https://doi.org/10.1145/280811.281008.

Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *CoRR*, abs/1902.04043, 2019. URL http://arxiv.org/abs/1902.04043.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3839–3848, Red Hook, NY, USA, 2018. Curran Associates Inc.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

L. S. Shapley. Stochastic games*. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953. doi: 10.1073/pnas.39.10.1095. URL https://www.pnas.org/doi/abs/10.1073/pnas.39.10.1095.

Jeremy Shen, Erdong Xiao, Yuchen Liu, and Chen Feng. A deep reinforcement learning environment for particle robot navigation and object manipulation, 2022. URL https://arxiv.org/abs/2203.06464.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. *ArXiv*, abs/2209.13085, 2022. URL https://api.semanticscholar.org/CorpusID:252545256.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning (1st ed.)*. MIT Press, Cambridge, MA, USA, 1998.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and YI WU. The surprising effectiveness of ppo in cooperative multi-agent games. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, page 24611–24624. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Appendix A

Additional Derivations

A.1 Lipschitz constant and relative observations.

Assuming a policy π, two agents in the scenario, and observations defined in eq. (3.15) as follows

$$\vec{o}_i' = \left[p_x^{(i)}, p_y^{(i)}, v_x^{(i)}, v_y^{(i)}\right] \bigoplus_{j \neq i} \left[p_x^{(j)} - p_x^{(i)}, p_y^{(j)} - p_y^{(i)}, v_x^{(j)} - v_x^{(i)}, v_y^{(j)} - v_y^{(i)}\right]$$

the 1-norm Lipschitz constant can be shown to depend only on the relative positions of the agents.

Representing this more compactly as:

$$\vec{o}_i' = \vec{o}_i \bigoplus_{j \neq i} \left[\vec{o}_i - \vec{o}_j\right]$$

We can find that this increases the effective Lipschitz constant by a constant factor.

$$\begin{aligned}
|\pi(o_1) - \pi(o_2)| &\leq K\ |o'_1 - o'_2| \\
&\leq K\ |\vec{o}_1 - \vec{o}_2| + |(\vec{o}_2 - \vec{o}_1) - (\vec{o}_1 - \vec{o}_2)| \\
&\leq K\ |\vec{o}_1 - \vec{o}_2| + 2|\vec{o}_2 - \vec{o}_1| \\
&\leq 3K\ |\vec{o}_1 - \vec{o}_2|
\end{aligned}$$

Increasing the number of agents would make this value rely on the relative positions of the other agents in the scenario. However, scenarios with more agents better communication strategies should be considered over appending relative observations.