GENEPI: a GENEric Parameter-sharing for Intrinsically motivated MARL agents

Joris Dinneweth
Univ Gustave Eiffel
TS2-MOSS
Champs-sur-Marne, France
Université Paris-Saclay
ENS Paris-Saclay, CNRS, SATIE
Gif-sur-Yvette, France
joris.dinneweth@univ-eiffel.fr

Abderrahmane Boubezoul
Univ. Gustave Eiffel
TS2-MOSS
Champs-sur-Marne, France
abderrahmane.boubezoul@univ-eiffel.fr

René Mandiau
Univ. Polytechnique Hauts-de-France,
CNRS, UMR 8201 - LAMIH
F-59313 Valenciennes, France
rene.mandiau@uphf.fr

Stéphane Espié
Univ. Gustave Eiffel
TS2-MOSS
Champs-sur-Marne, France
stephane.espie@univ-eiffel.fr

ABSTRACT

Multi-agent reinforcement learning (MARL) has become increasingly popular, but scaling to large populations of agents remains an open challenge. On the one hand, centralized training methods, such as parameter-sharing, address scalability issues when agents have homogeneous goals. On the other hand, only a few proposals concerned heterogeneous agents, which limits their scalability. This work extends parameter-sharing — an approach that factorizes multi-agent learning into a single shared policy — to agents with heterogeneous intrinsic motivations sampled from distributions. We focus our experiments on a traffic platform where human driver agents seek to optimize intrinsic motivations, namely, their own goals. In a context where decentralized baseline training failed to train ten agents, we show that our proposal allows scaling up to 90 heterogeneous agents on average in less than three minutes. Furthermore, our approach yields a generic learned policy that can adapt to new motivation distributions without retraining.

ACM Reference Format:
Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. 2023. GENEPI: a GENEric Parameter-sharing for Intrinsically motivated MARL agents. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 8 pages.

1 INTRODUCTION

Multi-agent systems (MAS) are a subfield of distributed artificial intelligence [21]. They constitute an opportunity to model the interactions between several autonomous entities with diverging information, diverging goals, or both. MAS have become increasingly popular in tackling real-world problems such as smart grids, the internet of things, robotics, and autonomous driving.

To solve complex problems, researchers often combine MAS with reinforcement learning (RL), a trial-and-error learning method for solving decision-making problems [23]. In multi-agent reinforcement learning (MARL), each agent interacts with an environment in which other agents can be considered as part [10]. Each agent aims to learn an optimal policy induced by feedback that rates its actions, deflects wrong ones, and reinforces good ones; hence, the name reinforcement learning. Despite growing interest, numerous challenges, such as non-stationarity, partial observability, and scalability, prevent MARL from becoming a go-to solution. For instance, scalability prevents large-scale applications involving more than a dozen agents, because each agent learns a policy via a neural network, which requires large amounts of computational resources and memory [16].

The two phases of the MARL learning process are training and execution. The two can be centralized or decentralized. Decentralization is considered the most realistic way to address real-world problems; however, centralization mitigates the MARL challenges in simulated environments. Parameter-sharing (PS) is one of the most popular centralization techniques. It learns a unique policy for all agents, reducing the parameter space by a factor of the number of agents [11]. However, having all agents share the same parameters can detrimentally affect learning [5]. To the best of our knowledge, there is no equivalent solution for agents with diverging goals. This issue prevents the large-scale application of heterogeneous agents in many real-world problems, such as research on future mobility, which studies the behavior of autonomous vehicles (AVs) cohabiting with human drivers in mixed traffic [2, 14]. The latter problem can be addressed by switching from extrinsic to intrinsic MARL reward functions [4]. The environment rewards extrinsically motivated agents, while intrinsically motivated agents reward themselves. Intrinsically motivated agents are often used to simulate human populations where each individual is self-interested. For instance, one may opt for intrinsically motivated agents (drivers) when attempting to reproduce human driver behavior in traffic. We based our work on the traffic case because we believe that intrinsically motivated MARL agents can compensate for the lack of "realistic"

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

human-driven vehicle models, as acknowledged by many authors [7].

This paper extends PS to heterogeneous intrinsically motivated agents. In our approach, heterogeneity comes from distributions of intrinsic motivations influencing agents' reward functions. We show that our resulting policy (1) supports large-scale application with up to 90 agents; (2) correctly discriminates and optimizes individualities induced by heterogeneous intrinsic motivations, and; (3) is generic, i.e., the policy successfully adapts to changes in the intrinsic motivation distribution without the need for retraining.

The remainder of this paper is organized as follows. Section 2 presents the MARL challenges and parameter-sharing. Section 3 details our proposal based on the GENEric Parameter-sharing for Intrinsically motivated agents (GENEPI). Then, Section 4 evaluates the results based on a road traffic platform and validates our approach. We also focus on scalability and genericity problems. Finally, Section 5 concludes the paper and provides perspectives.

2 BACKGROUND AND RELATED WORK

We briefly define MARL agents (2.1). Then, we introduce different MARL approaches into a taxonomy (2.2) and underline their main challenges (2.3). Finally, we focus on parameter-sharing (2.4).

2.1 Definition

Formally, MARL is defined as follows [10]. At regular time step t, a set of agents $\mathcal{I}$ observes the state $s_t \in \mathcal{S}$ of the environment and selects actions $a_{i,t} \in \mathcal{A}$. At the next time step $t+1$, the agents receive rewards $r_{i,t+1} \in \mathbb{R}$ according to their reward functions R_i, observe the next state $s_{t+1} \in \mathcal{S}$, and repeat the cycle (Figure 1a). For the sake of simplicity, we remove the agent notation i in all the following equations.

Depending on the application, MARL reward functions R can be extrinsic or intrinsic [4]. The environment rewards extrinsically motivated agents while intrinsically motivated agents reward themselves (Figure 1b). Often, one opts for intrinsically motivated agents when the underlying problem concerns simulating a self-interested (heterogeneous) human population.

Agents aim to play an optimal action selection policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ which provides the probability of selecting each action in a given state. An optimal policy π^* seeks to maximize the discounted sum of the rewards obtained over an episode, denoted by G_t (Equation 1),

$$G_t = \sum_{k=t+1} \gamma^k \cdot r_{t+k+1} \tag{1}$$

where the discount factor $\gamma \in [0,1)$ bounds this sum and is a trade-off between short and long-term rewards.

A state-action value function $Q^{\pi}(s,a)$ (Equation 2) estimates the expected future rewards and induces the agent's decisions towards an optimal policy $\pi^* = \max_a Q^{\pi}(s,a)$.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \mid \mathcal{S}_t = s, \mathcal{A}_t = a] \tag{2}$$

Since, in some environments, mapping the whole state-action space is intractable, MARL approximates the value function $Q(s,a)$ through a low-dimensional representation: a deep neural network.

2.2 MARL taxonomy

MARL algorithms fall into three categories depending on how agents learn the policy: value-based, policy-based, or actor-critic [23].

Value-based algorithms implicitly learn a policy by estimating the value of the actions $a \in \mathcal{A}$ in a given state $s \in \mathcal{S}$ according to the recursive Equation 3. The value of a state-action pair at the current time step t equals the sum of the reward obtained at t plus the discounted Q-value of the next time step $t+1$ if the agent acts optimally $a_{t+1} = \max_a Q(s_{t+1}, a_{t+1})$ according to its current estimation of $Q(s,a)$:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a_{t+1}) \tag{3}$$

The higher the value of $Q(s,a)$, the better the action. Exploring the entire state-action space would give the exact value of $Q(s,a)$ and thus the optimal policy, but this is impossible in large-scale environments and continuous state-action spaces. In the latter cases, the state-action value $Q(s,a)$ is approximated during the exploration process, which can bias its value when the exploration is insufficient. Some value-based algorithms save past experiences into a replay buffer, improving sample efficiency, i.e., the ability of an algorithm to get the most out of every sample.

Policy-based algorithms, instead, explicitly learn a policy that maps the probability distribution over possible actions $\pi : s \rightarrow a$. Consequently, an agent can learn a stochastic policy stronger than a deterministic one, especially in competitive multi-agents where deterministic behaviors are easily predictable. Nonetheless, these methods are less sample efficient and suffer from high variance because the mapping of π depends on which state-action the policy explores.

Actor-critic algorithms take advantage of the preceding algorithms while alleviating their weaknesses. A critic maintains a value approximation $Q(s,a)$ that appraises and induces the actor's decisions a. Numerous actor-critic algorithms have been proposed, such as A3C, DDPG, PPO, and TRPO [15, 17, 19, 20].

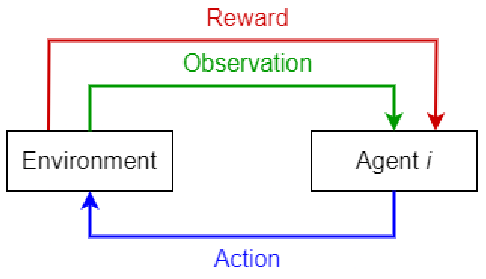
In this work, we choose D4PG, an actor-critic algorithm that has proven its efficiency in continuous control [1]. D4PG estimates $Q(s,a)$ via a distributional representation of the possible reward outcomes. In this algorithm, the critic stores transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in an experience replay buffer (ERB) to stabilize the learning process, whereas the actor explores the state space to map the environmental transitions better.

Regardless of the chosen algorithm, MARL raises different open challenges, as presented below.

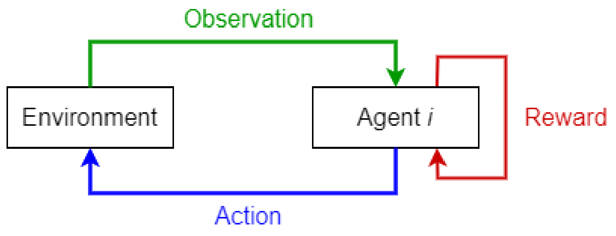
2.3 Challenges

Many challenges prevent MARL from being a go-to approach. Among the most cited in the literature, we list four: non-stationarity, curse of dimensionality, partial-observability, and heterogeneity.

Single-agent RL algorithms assume a Markovian environment, i.e., with a fixed state probability distribution [23]. However, MARL breaks this property because the agents continuously adapt to each other during the learning process. Consequently, the environment becomes non-stationary, which can hinder or prevent the convergence of the agents towards an optimal policy.



(a) Extrinsic rewards



(b) Intrinsic rewards

Figure 1: Reinforcement learning

To act optimally according to its goals, an agent should observe as much information as possible, including the entire environment and the agents' states. However, such information is rarely available in practice. Even if it were the case, processing this information with many agents would be too computationally costly because the state space grows exponentially with the number of agents.

To overcome the resulting curse of dimensionality, one can restrict the agents' observations to a reduced state space of the environment. This partial observability can be extended to the agent space so that agents can only observe some of their neighborhood. A designer must balance partial and full observability; otherwise, it will hinder either scalability or agent behavior.

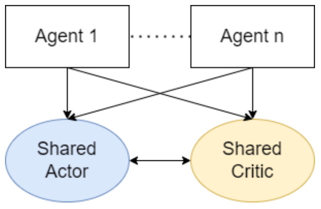
In MARL, the agents are either homogeneous or heterogeneous. Homogeneous agents are similar, share the same goals, and act in the same way. Conversely, heterogeneous agents may have different goals, knowledge, skills, or action spaces. This diversity impedes learning because agents must adapt to a broader range of possible behaviors and can rarely determine and anticipate their neighbors' goals.

Owing to these challenges, there is no convergence guarantee in MARL, except in simplistic scenarios [6]. Therefore, most studies narrow the number of learning agents to a dozen and empirically check for convergence [16]. Consequently, approaches such as parameter-sharing have been proposed to mitigate scalability issues.

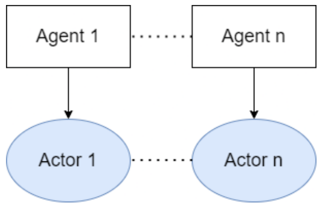
2.4 Parameter-sharing

Two phases compose MARL: training and execution; the two can be centralized or decentralized. Centralization often improves scalability by sharing extra information between agents, whereas decentralization requires agents to learn and act independently. *Lowe et al.* [16] introduced the centralized training decentralized execution (CTDE) method, which overcomes the shortcomings of the fully-centralized and fully-decentralized approaches. During the training phase, agents share additional information to alleviate scalability issues and discard them during the execution phase. The CTDE scheme includes two popular strategies, namely centralized critic decentralized actor (CCDA) and parameter-sharing (PS), that can be used depending on the nature of the agents.

Agents with heterogeneous goals can adopt a centralized critic decentralized actor (CCDA) scheme [16]. In CCDA, all agents feed a central critic with extra information to improve the assessments



(a) Centralized training



(b) Decentralized execution

Figure 2: Parameter-sharing with actor-critic

of actors. During execution, the critic is discarded; then, the agents act in a decentralized way.

When agents are homogeneous, i.e., share common goals, one can opt for parameter-sharing (PS) [11]. PS mitigates the curse of dimensionality by centralizing agents' experiences and factorizing policy training by using a unique shared neural network. In other words, PS reduces the parameter space by a factor of the number of agents $|\mathcal{I}|$, which lowers the training time to tractable levels and alleviates scalability issues. Once learned, each agent receives a copy of the learned policy and acts in a decentralized way (Figure 2).

PS was initially designed to optimize the learning of homogeneous agents with the same reward functions. However, only a few authors have attempted to extend PS to agents with heterogeneous goals [5, 12, 24]. Here, we introduce the closest attempts we found on *Scopus*, *Web of Science*, and *IEEE Xplore* using the keywords *multi-agent reinforcement learning*, *parameter-sharing*, and *heterogeneity*.

The first attempt to extend PS to heterogeneous agents was by overloading the observation space with a binary index indicating which of the two goals between lane-following and overtaking, an agent must optimize [12]. However, the binary index prevents agents from simultaneously optimizing more than two goals, thereby diminishing heterogeneity. The authors trained 6 agents and performed tests with up to 15 agents. For lane-following, collisions were observed approximately 11% times.

The second method learns multiple PS policies and partitions agents into roles according to their goals, where a role matches a PS network [5]. Since this method automatically determines the number of roles, i.e., the number of neural networks, the parameter space can grow and impede scalability. The authors trained up to 200 agents partitioned into 4 different roles.

The third method learned an actor-critic algorithm. Agents share the same critic, but each learns a policy to prevent the agent's personality from being absorbed by a shared policy [24]. Although this approach prevents PS from reducing the variance between agent policies, it is still poorly scalable. Additionally, when one wants to add more agents with different goals, this framework requires retraining, because each agent learns a neural network. The authors simultaneously trained up to 15 agents.

By reducing heterogeneity or scalability, none of these extensions equals the original PS performance for agents with heterogeneous goals. Thus, we propose an alternative extension of PS to overcome the scalability issue for intrinsically motivated agents. Our proposal neither impedes scalability nor heterogeneity compared to the reviewed approaches.

3 OUR APPROACH: GENEPI

In this section, we present our approach called GENEPI. Thus, we describe the global view in (3.1). From this global view, we then characterize three main practical elements: (i) the context of the traffic environment (3.2), (ii) the neighborhood (3.3), and (iii) intrinsic motivations (3.4).

3.1 Global view

Recall that the core objective of our proposal is to learn a GENEric Parameter-sharing for intrinsically motivated agents (GENEPI). Figure 3 illustrates the proposed approach. It consists of two steps: (i) initialization and (ii) training.

At the initialization step, agents (Agent 1, . . ., Agent N) draw motivations that determine their reward functions. The first motivation encourages agents to reach their desired speed, whereas the second prevents them from taking high risks. We further detail these Gaussian motivation distributions in Section 3.4.

During the training (second and last step), agents observe the environment, build their ego-centered observations, and process them to the shared policy. Then, agents act according to the actions returned by the shared policy and reward themselves depending on their new observations and intrinsic reward functions. The shared policy produces different output behaviors when it appropriately learns to discriminate between agents' intrinsic motivations passed as input.

Thus, GENEPI takes three types of inputs: (1) environment from the agent's viewpoint, (2) neighboring agents from an agent's viewpoint, and (3) motivations influencing agents' reward functions.

3.2 Our context: traffic environment based on ArchiSim

We base our agent on the ArchiSim traffic simulator [8], which relies on psychologist studies of drivers' behaviors [9]. ArchiSim divides the road into control zones, whose width adapts to the velocity of the driver to ease the reproduction of complex behaviors and, more specifically, to anticipate ongoing flow variations. A zone indicates the average speed of the drivers within or the speed limit when it is empty.

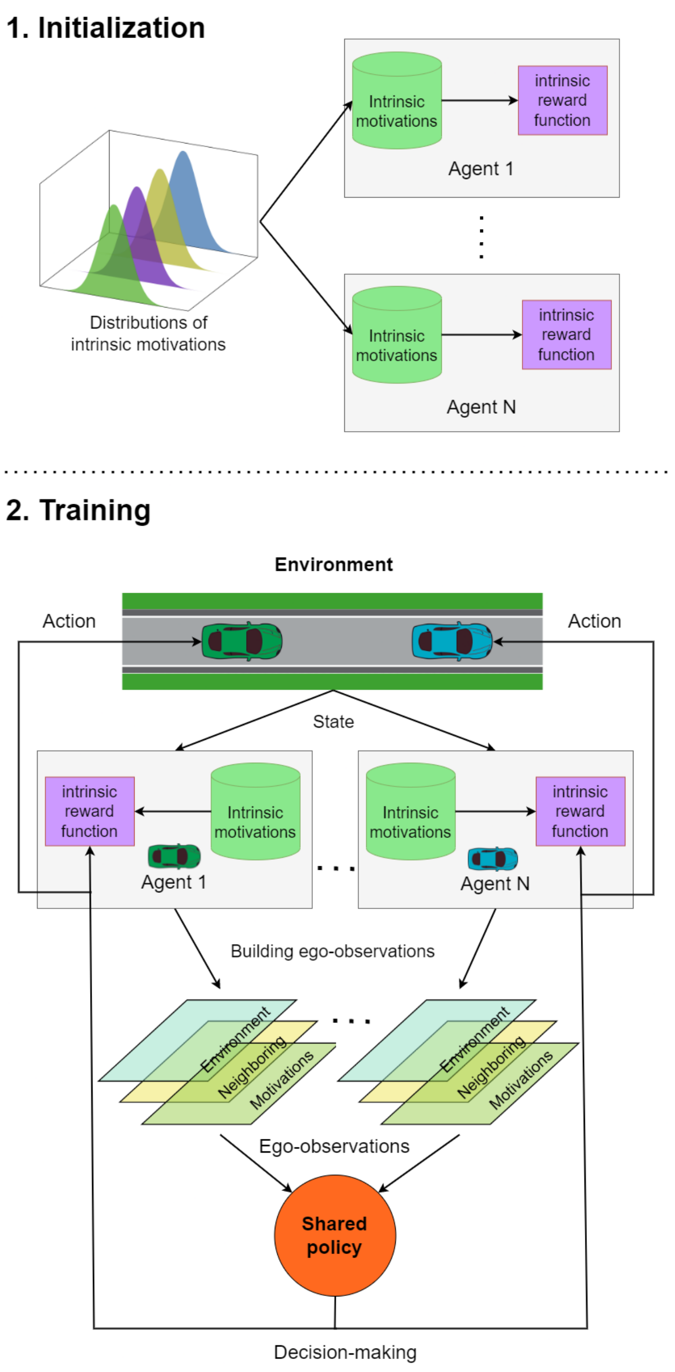


Figure 3: Initialization and training of GENEPI

In our case, ego-centered observations of the environment include the control areas of the driving agent's current lane because we only consider longitudinal driving in this straightforward problem, as Figure 4 illustrates. Each area, whose width depends on the ego-vehicle (in red) velocity, aggregates data and computes the average speed of the vehicles within.

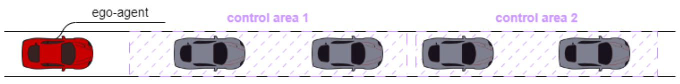


Figure 4: Control areas from an agent's perspective

In our experiments, agents aim to drive along a road while accumulating rewards. We set random initial velocities $v_i^{init} \in [28, 36]\,m/s$ at the beginning of each episode. The road is two kilometers long, which is sufficient to observe behavioral changes. Note that the more vehicles that train simultaneously, the less distance they will have to travel on average.

3.3 Neighborhood

Observing the environment partially improves the scalability. Additionally, in our application, we assume that it is also a "more realistic way" to reproduce the concept of bounded rationality [18], the fact that humans can be distracted and lack information and time to act optimally. A partial observation includes the time headway and time to collision of the two preceding vehicles from the vehicle in front of them (Figure 5). The four resulting variables help an agent forecast the behavioral changes of the preceding vehicles induced by incoming traffic conditions.

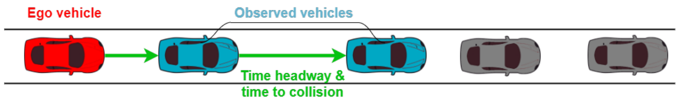


Figure 5: Observation of the neighborhood

3.4 Intrinsic motivations

As the introduction states, we aim to simulate intrinsically motivated driver agents. Starting from a straightforward example, in which agents can only move longitudinally, we assume that drivers would like to reach a compromise between (i) driving at their velocity R_{velocity} and (ii) maintaining safe distances R_{safe}. An agent can only accelerate or decelerate via continuous control a.

The first reward R_{velocity} encourages an agent to reach its desired speed v_i^* (Equation 4). An agent obtains the maximal reward when its current speed v_i equals its desired speed v_i^*. An agent computes its desired velocity as the deviation from the speed limit of the road drawn from a Gaussian distribution: $v_i^* = \text{speed limit} \times \mathcal{N}(1, 0.03)$.

$$R_{\text{velocity}} = \left(\frac{\min(v_i, v_i^*)}{\max(v_i, v_i^*)}\right)^2 \tag{4}$$

The second reward function R_{safe} discourages an agent i from closing with the leader vehicle j (Equation 5). The function punishes an agent i when the evaluated risk $ER(i, j)$ exceeds its accepted risk threshold δ_i, drawn from a Gaussian distribution $\delta_i \sim \mathcal{N}(1.5, 2)$. We consider that $ER(i, j)$ (Equation 6) comes from a previous study initially proposed by [13]. This author identified which time headway $\Delta_t(i, j)$ and time to collision $\Delta_{TTC}(i, j)$ trigger the brake decision in car-following situations.

$$R_{\text{safe}} = \begin{cases} \delta_i - ER(i, j) & \text{if } ER(i, j) > \delta_i \\ 0 & \text{else} \end{cases} \tag{5}$$

$$ER(i, j) = \left(\frac{1}{\Delta_t(i, j)} + \frac{4}{\Delta_{TTC}(i, j)}\right) \tag{6}$$

Finally, we weight both reward outcomes equally $R = R_{\text{safe}} + R_{\text{velocity}}$.

4 EXPERIMENTS AND RESULTS

We present three criteria for the evaluation of GENEPI (4.1) and their results (4.2).

4.1 Evaluation of experiments

We perform experiments with the D4PG algorithm to assess three criteria of our approach: scalability, diversity, and genericity. These three criteria are essential for validating the advantages of GENEPI compared to a decentralized baseline [3]. The only difference from the baseline is that GENEPI adopts parameter-sharing.

Experiment 1 *based on the scalability criterion.* In the first experiment, we compare the convergence time of GENEPI with a decentralized training approach for different numbers of agents. We expect GENEPI to outperform the decentralized baseline.

Experiment 2 *based on the heterogeneity criterion.* In the second experiment, we ensure that heterogeneity is preserved alongside the use of PS. We expect that the GENEPI policy network will correctly discriminate agents' intrinsic motivations and learn appropriate behaviors accordingly.

Experiment 3 *based on the genericity criterion.* In the third experiment, we check the generic property of GENEPI by executing the learned policy with a distinct motivation distribution from the learned policy. We expect our agents to act appropriately in an unknown motivation distribution.

Table 1 shows the hyperparameters of these experiments. Note that we use a larger learning rate in our framework because the diversity of experiences brought by the agents during the training stabilizes the learned policy. The size of the replay buffer is sufficiently large to contain transitions (s_t, a_t, r_t, s_{t+1}) from different training stages, which helps map the state-action space with rewards in the long term. Among the few neural network architectures that have been specifically studied for reinforcement learning, we chose the D2RL architecture, which is derived from *Sinha et al.* [22], who identified an appropriate network architecture that performs well in RL problems.

4.2 Results

This subsection details the scalability, heterogeneity, and genericity results. Subsequent results are obtained using an *NVIDIA RTX 3080 TI.*

Scalability. We evaluate the scalability of a decentralized baseline and GENEPI. Thus, for these two approaches, we investigate 3, 6, 10, 30, 60, and 90 learning agents. We check convergence every minute. When all the agents accumulate at least 90% of the rewards

Table 1: Hyperparameters

Parameter	Value
Optimizer	Adam
Learning rate (GENEPI)	$1e-3$
Learning rate (decentralized)	$3e-4$
Buffer size	$5e+4$
Discount factor γ	0.85
Batch size	64
Neural architecture	D2RL

over an episode, we consider that they converge. This score implies that no accidents occurred during the testing phase. After twenty hours of training without convergence, we halt the learning process and assume that the agents fail to converge. Thus, each experiment is (randomly) repeated five times, with different neural network initializations and vehicle starting positions. We then measure the average convergence time and its standard deviation (Table 2).

Table 2: Average duration and deviation (in minutes) for the training until convergence. The * symbol denotes the non-convergence

# agents	GENEPI	Baseline
3	**3.4 ± 1.5**	14 ± 5
6	**2.6 ± 0.5**	28 ± 26
10	**2 ± 0.6**	*
30	**1.6 ± 0.8**	*
60	**2.2 ± 1.2**	*
90	**2.6 ± 0.8**	*

The results show that the decentralized baseline fails to converge in less than twenty hours when more than six agents learn. We attribute this failure to non-stationarity problems since, in our scenario, agents struggle to adapt to the behaviors of other vehicles, which are continually changing during the learning process.

As expected, the GENEPI approach converges faster than the decentralized baseline. GENEPI's convergence time remains almost constant (2.4 minutes on average), regardless of the number of learning agents. This consistency is probably a consequence of using PS, which keeps the parameter space size constant, whatever the number of agents trained. We do not test whether this consistency remains valid for a more significant number of agents. Nevertheless, conducting more than five tests could refine this average convergence time.

Gupta [11] observed that PS provided a natural curriculum for cooperative agents. Our results show that this observation is also valid in a mixed setting. In decentralized training, each agent acts according to a randomly initialized neural network, which leads to a broader heterogeneity of behaviors among agents at the beginning of training. Conversely, at initialization, agents act homogeneously with PS because they share the same parameters, providing a natural curriculum. We can observe this phenomenon on the convergence of GENEPI against the baseline (Figure 6). Before the training, at

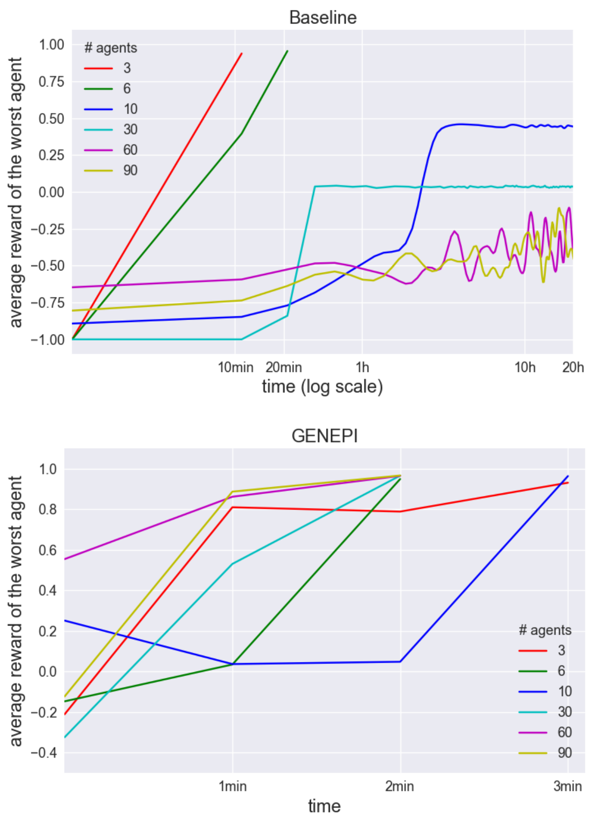


Figure 6: Convergence time of the baseline against GENEPI

the initialization (time=0s), the worst average reward that agents acting through GENEPI receive is greater than -0.4, while the value down to -1 for the baseline.

In the decentralized approach, agents strive to adapt to constantly changing self-interested behaviors, whereas, in the centralized method, they learn to generalize over a possible set of behaviors.

Now that we have shown the scalability of GENEPI against the decentralized baseline, we narrow the number of agents to five in the following experiments to provide more straightforward examples of GENEPI's heterogeneity and genericity.

Heterogeneity. To assess the behavioral heterogeneity of our PS agents, we track the evaluated risk taken over a simulation for all agents. We endow each agent with a higher preferred speed than the one in the preceding position, forcing the agents to reach their risk threshold to accumulate maximum rewards.

Figure 7 shows the estimated risk four agents took over time. Agents 0, 1 and 2 reach their risk thresholds, respectively $\delta_0 = 2$, $\delta_1 = 1.8$, $\delta_2 = 1.6$, whereas Agent 3 is still reaching its threshold ($\delta_3 = 1.4$). The PS policy correctly differentiates agents' motivations and learns diverse behaviors accordingly. The results confirm that PS can be applied to intrinsically motivated heterogeneous agents.

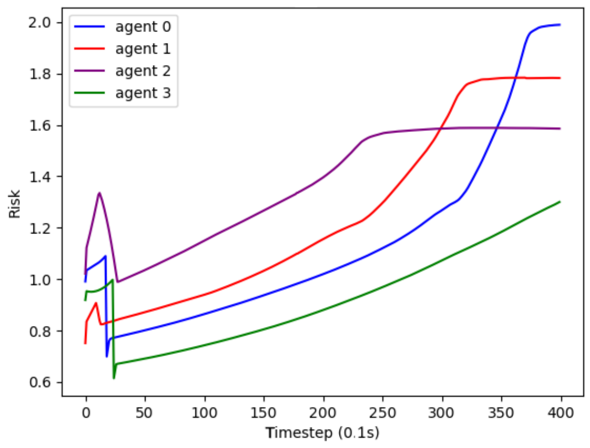


Figure 7: Evaluated risk over time of four vehicles

Genericity. Genericity refers to the ability of our learned policy to perform well when simulating agents with intrinsic motivations that differ from those learned. In other words, the agents' behaviors can be changed without retraining the network.

Initially, the agents learn their accepted risk threshold $\delta_i \in [1, 2]$. When we change this distribution to $\delta_i \in [2, 3]$ without retraining the policy and observe that agents behave correctly according to the new distribution and reach a risk threshold in the range $[2, 3]$ instead of the $[1, 2]$ learned range (Figure 8).

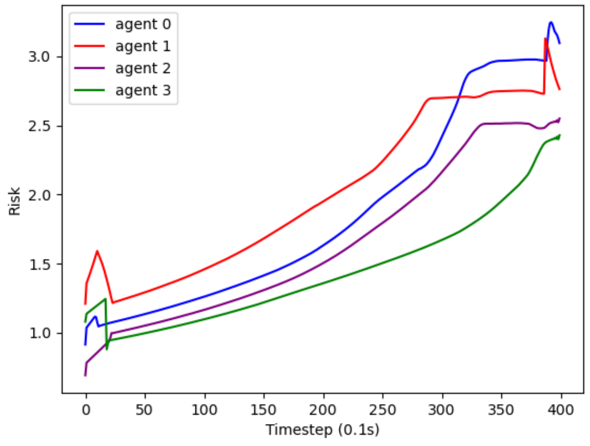


Figure 8: Evaluated risk over time without retraining

As expected, GENEPI is scalable, generic, and preserves heterogeneity.

5 CONCLUSION AND DISCUSSION

This work proposes an approach to mitigate scalability issues in multi-agent reinforcement learning (MARL) with heterogeneous intrinsic motivations. This approach extends parameter-sharing, a centralized training approach that learns a shared policy for homogeneous agents, to a heterogeneous case. We call our method GENEPI for GENEric Parameter-sharing for intrinsically motivated agents.

We compare GENEPI's scalability with a decentralized baseline for a simple road traffic use case. While the latter fails to converge when ten agents simultaneously learn, GENEPI successfully scales up to 90 agents in less than three minutes, implying that GENEPI correctly discriminates agents' intrinsic motivations within a shared policy and that no accidents occurred. Moreover, we show the genericity of our method, meaning that we can train and test a shared policy on distinct motivation distributions without negatively affecting agents' behaviors.

As mentioned earlier, the only limitation of our proposal is that it only applies to intrinsically motivated agents, which typically reflects applications where one wants to simulate human populations, such as drivers, cyclists, and pedestrians. Thus, our proposal can contribute to simulating large populations of intrinsically motivated human drivers, which the literature needs to improve further.

In future work, we plan to assess the robustness of GENEPI in more complex traffic scenarios involving additional intrinsic motivations that not all agents would share. For instance, a merging scenario in which some drivers yield while others do not.

REFERENCES

[1] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributed distributional deterministic policy gradients.

[2] Ana LC Bazzan and Franziska Klügl. 2009. *Multi-agent systems for traffic and transportation engineering.* IGI Global.

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

[4] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2004. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou (Eds.), Vol. 17. MIT Press, Vancouver.

[5] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. 2021. Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, Virtual Event, 1989–1998. https://proceedings.mlr.press/v139/christianos21a.html

[6] Kai Cui, Anam Tahir, Gizem Ekinci, Ahmed Elshamanhory, Yannick Eich, Mengguang Li, and Heinz Koeppl. 2022. A Survey on Large-Population Systems and Scalable Multi-Agent Reinforcement Learning. *ArXiv* abs/2209.03859 (2022).

[7] Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. 2022. Multi-agent reinforcement learning for autonomous vehicles: a survey. *Autonomous Intelligent Systems* 2, 1 (2022), 1–12. https://doi.org/10.1007/s43684-022-00045-z

[8] Arnaud Doniec, René Mandiau, Sylvain Piechowiak, and Stéphane Espié. 2008. A behavioral multi-agent model for road traffic simulation. *Engineering Applications of Artificial Intelligence* 21, 8 (2008), 1443–1454.

[9] Stéphane Espié and Farida Saad. 2000. Driver behaviour modelling and traffic simulation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 44, 20 (2000), 3–251–3–254.

[10] Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review* 55, 2 (2022), 895–943. https://doi.org/10.1007/s10462-021-09996-w

[11] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*, Gita Sukthankar and Juan A. Rodriguez-Aguilar (Eds.). Springer International Publishing, Cham, 66–83.

[12] Meha Kaushik, Nirvan Singhania, and K Madhava Krishna. 2019. Parameter sharing reinforcement learning architecture for multi agent driving. In *Proceedings of the Advances in Robotics 2019*. Association for Computing Machinery, Kaiserslautern, 1–7. https://doi.org/10.1145/3352593.3352625

[13] Takayuki Kondoh, Tomohiro Yamamura, Satoshi Kitazaki, Nobuyuki Kuge, and Erwin Roeland Boer. 2008. Identification of visual cues and quantification of drivers' perception of proximity risk to the lead vehicle in car-following situations. *Journal of Mechanical Systems for Transportation and Logistics* 1, 2 (2008), 170–180. https://doi.org/10.1299/jmtl.1.170

[14] Changjian Li and Krzysztof Czarnecki. 2019. Urban Driving with Multi-Objective Deep Reinforcement Learning. In *AAMAS '19: Proceedings of the 18th international conference on multiagent systems*. 359–367.

[15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[16] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments, In Proceedings of the 31st International Conference on Neural Information Processing Systems. *NeurIPS* 30, 6382–6393. https://doi.org/10.5555/3295222.3295385

[17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[18] Christophe Mundutéguy and Isabelle Ragot-Court. 2011. A Contribution to Situation Awareness Analysis: Understanding how mismatched expectations affect road safety. *Human factors* 53, 6 (2011), 687–702. https://doi.org/10.1177/0018720811420841

[19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

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

[21] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

[22] Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. 2020. D2rl: Deep dense architectures in reinforcement learning.

[23] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press.

[24] Ning Yang, Bo Ding, PeiChang Shi, and Dawei Feng. 2022. Improving scalability of multi-agent reinforcement learning with parameters sharing. In *2022 IEEE International Conference on Joint Cloud Computing (JCC)*. IEEE, 37–42. https://doi.org/10.1109/JCC56315.2022.00013