Impact of Heterogeneity on Multi-Agent Reinforcement Learning

Rodrigo Fonseca Marques[1], **Zenilton Kleber Gonçalves do Patrocínio Júnior**[1]

[1]Image and Multimedia Data Science Laboratory (IMSCIENCE),
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, Minas Gerais, Brazil

`marquesfonsecarodrigo@gmail.com, zenilton@pucminas.br`

Abstract. *Most Multi-Agent Reinforcement Learning (MARL) methods and studies use homogenous agents. The majority of study on heterogeneity concentrates on agents with different skill sets. However, in real-world applications, agents frequently possess the same set of skills but different degrees. In this paper, we propose a novel model for heterogeneous agents in a MARL system, in which they share a standard skill set but have different degrees of intensity. Experiments were carried out in the framework of Soccer Twos, a competitive and cooperative game, and also with Tennis, which has competitive gameplay. Results demonstrate that heterogeneous agents perform better than homogeneous ones in both environments and also acquire organizational abilities in Soccer Twos.*

1. Introduction

Recently, reinforcement learning (RL) has been used more frequently to train autonomous agents to solve difficult tasks. Examples of problems in which RL has been successfully applied are playing video games at a super-human level in Atari [Mnih et al. 2015], in Chess [Silver et al. 2018], in Go [Silver et al. 2016], but also in robotics [Kober et al. 2013], in routing [Nazari et al. 2018] or in chronic illness treatment [Shortreed et al. 2011], among others.

RL has been used a lot with single agents tasks, but many situations require the employment of multiple agents. A multi-agent system consists of a group of autonomous agents interacting in the same environment. Those agents observe the environment with their sensors and act with their actuators [Weiss 1999, Vlassis 2022]. A multi-agent reinforcement learning (MARL) is applied in that scenario.

In a MARL system, a set of agents interacts with a common environment. And, at each time step, each agent performs an action to achieve its goal [Oroojlooy and Hajinezhad 2023]. These multiple agents can have a common or even an opposite goal, cooperating with one another or competing against each other.

Most of the works in MARL systems have homogeneous agents, meaning that they are identical to each other. However, in real-world applications with multiple agents in the same environment, these agents may have heterogeneous structures [Oroojlooy and Hajinezhad 2023, Wakilpoor et al. 2020].

In this paper, we propose a new model for heterogeneous agents on multi-agent reinforcement learning. In our proposal, agents have the same set of abilities/skills but with different intensities. Here, an ability/skill set for an agent represents how it can observe and act on the environment. Therefore, in our model, all agents possess the same sensors but with different acuity levels, and they also have the same actuators but also with distinct control levels. Thus, although they have the same ability set, agents can excel in some assignments because of the different intensity levels they present on those skills. And that is a similar situation to a real-world scenario.

Experimental results show that the proposed model trained with heterogeneous agents achieved higher scores compared to the use of homogeneous agents. In both environments, Tennis and the Soccer Twos games from Unity ML-Agents Toolkit [Juliani et al. 2018], the heterogeneous agents behaved differently compared to the baseline with homogeneous ones. It was clear in the Soccer Twos that the heterogeneous agents have learned how to organize themselves, one of them defending while the other attacking. That was possibly caused because each agent possessed the best skill set to perform in that role. With agents that are different from each other, more distinct experiences should be possible, and, with that, the agents can learn more.

The rest of the paper is organized as follows. Section 2 presents some related works. Section 3 introduces the terminology and some useful notations. In Section 4, we present the proposal as well as the mathematical model for heterogeneous agents. Section 5 explains the experimental setup and the obtained results. Finally, we draw some conclusions and possible future works in Section 6.

2. Related Works

In MARL, an issue much discussed is the cooperation among the agents to maximize the total reward. In [Li et al. 2020], the authors studied cooperative MARL with hierarchical relation graphs in a partially observable game. They proposed a new method, called *Hierarchical Relation Graph Soft Actor-Critic* (HRG-SAC), which explores spatial relationships. In the first step, they used a hierarchical graph generation to represent the spatial relationships among the agents. After that, they utilized some graph information as input for a convolution network, and finally, they applied the *Soft Actor-Critic* (SAC) algorithm to train their agents. Then during training, they used the Food Collector game from the Unity ML-Agents Toolkit (which is the same platform used in this work). Experimental results showed a significant improvement with their approach.

A MARL system is utilized in [Wakilpoor et al. 2020] to map an unknown environment through heterogeneous agents. Their approach is called *Embedded Multi-agent Actor-Critic* (EMAC), an algorithm for training decentralized policies in multi-agent scenarios for both homogeneous and heterogeneous agents. Experimental results showed that EMAC outperforms traditional multi-agent coverage path planning techniques. The method was tested in a cooperative system. Besides, the robustness and scalability of the number of agents in EMAC were evaluated, and the final results indicated that it scales well.

In [Fu et al. 2022], the authors also use heterogeneous agents in a cooperative scenario. Their proposal is a *Heterogeneous League Training* (HTL), an efficient algorithm for heterogeneous cooperation systems. The method uses heterogeneous teammates with different cooperation skills to achieve robust agent policies. Their approach, HTL, improves the performance of the agents compared with other existing methods.

In [Bowling and Veloso 2000], there is an in-depth review and analysis of stochastic games (SG) theory for MARL. In this work, the authors studied SG, and also they discussed the main difference between them and the Markov decision process (MDP). It is worth mentioning that they also consider both the communities of game theory and RL communities accountable for presenting relevant techniques to be applied with MARL systems.

A cooperative and competitive environment is used in [Liu et al. 2019]. The authors also use a 2v2 soccer game using the MuJoCo physics engine to demonstrate that decentralized training can lead the agents from ball chasing to evidence of cooperation.

The main difference between this work and all the previous works presented in [Wakilpoor et al. 2020, Fu et al. 2022, Bowling and Veloso 2000, Liu et al. 2019] is that they consider the agents to have different skill sets, meaning the agents have different abilities from each other, while in this work, all agents have the same set of skills but with different levels of intensity. And finally, in [Liu et al. 2019] all the agents are equal to each other, with homogeneous skill sets, that differ from our study that measure the heterogeneity in each skill.

One of the advantages of having several agents with the same skills, but different intensities, is that this type of approach is the closest we have to real life. Furthermore, in certain environments, it is not possible for agents to have different abilities. With that, the proposed approach brings a new possibility to improve the training of multiple agents.

3. Background

3.1. Markov Decision Process

In RL, the Markov decision process (MDP) can model the iterations between an agent and its environment. Thus it can represent the agent's decision-making according to its possible states, actions, and rewards.

RL agents explore their environment trying to maximize a reward signal. A reward is generated by taking an action in a certain state. Thus, an agent can make new decisions (based on past experiences), trying to maximize its future reward. The value function represents the expected total discount reward to be received in the current state when following policy. Moreover, an action policy is the set of rules or possible behavior that the agent can take in a given state. And, finally, the optimal policy represents the best action to take in the given state that allows the agent to maximize its reward.

An MDP iteration starts when the agent, in a given state S_t, takes an action A_t, which modifies the environment (in which it finds itself). Once the environment has changed, the agent observes the newly generated state S_{t+1} and receives a reward r_{t+1}. As the agent then receives this new value, the new state becomes the current state. So, the agent takes another action, restarting the cycle [Sutton and Barto 2018].

3.2. Stocastic Games

A stochastic game (SG) can be considered as a generalization of an MDP to a multi-agent scenario. An SG is very similar to the MDP framework. But in SG, multiple agents perform actions, and the received reward and next state jointly depend on the actions performed by all the agents [Busoniu et al. 2008, Lowe et al. 2017].

An SG is a tuple $< N, S, A, O, P, R, \gamma >$, in which N is the number of agents, S is a set of states, the set of actions is represented by $A_1, ..., A_N$, and the set of observations $O_1, ..., O_N$ for each agent. P represents the transition probability between the states, R is the reward function, and γ is the discount factor.

At each instant of time t, each agent i observes the environment state S_t and, according to its policy π_i, performs the action a_i^t and receives a reward R_i^t. Here, we consider that the agents are in a partially observable system, so each of them has access only to its local observation o_i^t, as well as to its separate reward [Oroojlooy and Hajinezhad 2023, Bowling and Veloso 2000].

The observation that each agent i makes is defined by the observation function o_i given by Equation 1, while Equation 2 presents its stochastic policy function π_{Θ_i} that it uses to choose the action to be taken.

$$o_i \;:\; S \rightarrow O_i \tag{1}$$

$$\pi_{\Theta_i} \;:\; O_i \times A_i \rightarrow [0,1] \tag{2}$$

The state transition function P is given by Equation 3, and it is worth mentioning again that a new state depends on the set of actions performed by all agents.

$$P : S \times A_1 \times ... \times A_N \times S \rightarrow [0,1] \tag{3}$$

Each agent i receives a reward R_i according to the action performed given by Equation 4. And, the final goal of each agent i is to maximize its own total expected return G_i given by Equation 5 [Sutton and Barto 2018].

$$R_i : S \times A_i \times ... \times A_N \times S \rightarrow \mathbb{R} \tag{4}$$

$$G_i = \sum_{t=0}^{T} \gamma^t R_i^t \tag{5}$$

Finally, SGs are not just an extension of an MDP with multiple agents. They can be considered as an extension of matrix games with multiple states. It is possible to see each state of an SG as a matrix game [Bowling and Veloso 2000].

4. Model for Heterogeneous MARL

In this work, we propose a model for heterogeneous agents that can observe the environment with distinct acuity levels and also act on it with different intensity magnitudes, making the agents heterogeneous from each other. In real-world scenarios, one should expect that each agent has its level of expertise in the way that it interacts with the environment. Thus, some agents may perform better than others in specific assignments.

In a parallel comparison with the real-world environment, in a Basketball game, for example, all players can run, jump, pass, and shoot. Although all players have the same skill set, they do not have the same intensity. Some players are faster than others,

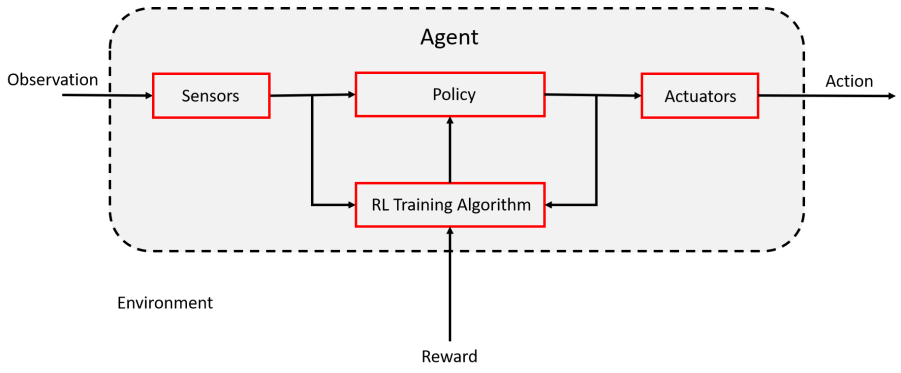


Figure 1. Detailed outline of an agent structure.

stronger, and some of them have a better peripherical vision. This imbalance between the skills of the players and knowing what each of the players is good at, and taking advantage of his skill, is what makes the team and the players be better.

The strategy we focus on to make the agents heterogeneous is varying sensing and actuation components from each agent, allowing them to perceive and act differently from others. Figure 1 represents a detailed outline of an agent structure used in our MARL system. An agent possesses sensors that convert stimuli into percepts and actuators that convert commands into actions. Each agent uses its sensors to observe the environment. And, each of those observations is used to update its policy (e.g., through the RL training algorithm). Then, according to that policy, it generates a command that is translated into action through its actuators. And, in the end, this process ends with the agent receiving a reward for the chosen action.

We adopt a *filtering strategy* to intensify or attenuate stimuli perception and thus to model all agents with the same set of sensors but with different acuity levels. Note that each game has its specificities and set of abilities to play, so each agent has different skills in each environment. Therefore, an agent observation passes through a filter before it is actually used by it, and Equation 6 represents that step, in the specific environment. Similarly, different magnitudes of action are generated by the same set of actuators using an analogous principle. Thus, the same command may result in different action magnitudes depending on the filter used. That is represented by Equation 7. Without loss of generality, the same set of sensors (or actuators) could be considered for every agent because a specific sensor (or actuator) can be eliminated entirely from the agent structure through its full filtering (i.e., setting its output to zero).

$$\Psi_i : O_i \to O'_i \tag{6}$$

$$\Phi_i : A_i \to A'_i \tag{7}$$

The Ψ and Φ functions are used to vary the agent's observations and actuators' intensity. For example, it is possible to apply a reduction, or attenuation of the agent observation, by changing the value of the Ψ function and applying it to the agent's specific skill. The same can be done with any skill.

Similar to Equation 2, each heterogeneous agent chooses an action to take considering now the filtered observation O'_i and using its new stochastic policy given by

Equation 8. And, the transition function depends on the filtered actions A_i's of all agents as given by Equation 9.

$$\pi_{\Theta_i} : O_i' \times A_i \rightarrow [0, 1] \tag{8}$$

$$P : S \times A_1' \times ... \times A_N' \times S \rightarrow [0, 1] \tag{9}$$

Finally, agent reward also depends on all filtered actions A_i's and is given by Equation 10.

$$R_i : S \times A_i' \times ... \times A_N' \times S \rightarrow \mathbb{R} \tag{10}$$

The use of mappings Ψ_i and Φ_i allows agents to be different from each other, and therefore it is able to represent a scenario with heterogeneous agents. Moreover, those mappings are very flexible and allow different intensification and attenuation schemes.

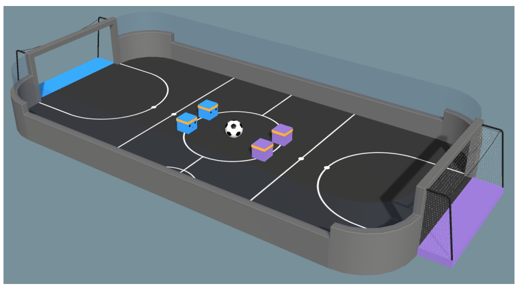
5. Experimental Results

5.1. Test Environment

For the experiments, we used multi-agent environments called Soccer Twos and Tennis provided by the Unity platform in the Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) [Juliani et al. 2018]. This toolkit is an open-source platform in which users can simulate and train RL agents or even create their own game environment.

The Soccer Twos game environment is a two-versus-two soccer game with two blue agents against two purple agents. These agents are in the same field. The blue team is composed of Agents 1 and 3, and it is opposed by the purple team, composed of Agents 2 and 4, as shown in Figure 2(a). It is a competitive and cooperative game in which the objective of each team is to get the ball into the opponent's goal while preventing the ball from entering its own goal.

The second environment is the Tennis game, it is a one-versus-one game. Each agent has its own side of the field. Their goal is to make the ball fall into the opponent's side. Figure 2(b) shows the field of the Tennis game. The game starts with the ball randomly falling from a distance into one side, and then the agent must prevent it from happening on their side of the field hitting the ball to the other side.


(a) Soccer Twos

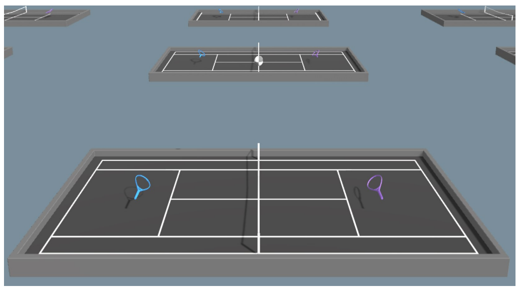

(b) Tennis

Figure 2. Game environments.

In Unity ML-Agents Toolkit, agents can perceive the environment through numeric or visual observations. Numeric observations measure attributes of the environment from the point of view of the agent. While visual observations, on the other hand, are images generated from the cameras attached to the agent and represent what the agent sees at that given moment. It is common to confuse an agent's observation with the environment (or game) state. The environment state represents information about the entire scene containing all the game characters. However, the agent observation contains only information that it is aware of and is typically a subset of the environment state. In regard to actions, they can either be continuous or discrete depending on the complexity of the environment and agent.

All agents in the Soccer Twos game have their vector of numeric observation and some discrete actions. The 336-D observation vector corresponds to 11 ray casts forward distributed over 120 degrees and 03 ray casts backward distributed over 90 degrees. Each ray detects 06 possible object types, along with the object's distance. The forward ray casts contribute to 264 dimensions and backward ones for the remaining 72 dimensions. The discrete actions correspond to forward, backward, and sideways movement, as well as rotation.

In the Tennis game, each agent has a 24-D observation vector corresponding to the position, velocity, and orientation of the ball and the racket. And they can control the movement of the racket towards the net and its rotation using a 2-D continuous vector.

5.2. Heterogeneous Agent Classes

To assess the impact of heterogeneous agents in a MARL system, we define observation classes $\Psi = \{\Psi_0, \Psi_1, \Psi_2\}$, according to some sensing traits that may be modified – see Table 1. Since a Soccer Twos agent observes the environment using ray casting, the following sensing traits were selected to be changed: (i) *'Max Ray Degrees'*; and (ii) *'Ray Length'*. They represent the angle (in degrees) at which rays spread out and the maximum distance a ray goes for detection, respectively. Similarly, in the Tennis game, the traits selected were: (i) *'Ball Visual'*; and (ii) *'Agent Visual'*. Representing the perception that the agent has of the ball in the field and the other one, the vision of where its opponent is respectively. By changing the agent's visual perception, it's like we limiting the agent's ability to know exactly where his opponent or the ball is. Table 1 shows the changes in the sensing traits for each observation class considered in our experiments, in which observation class Ψ_0 represents the baseline, i.e., no change.

The intensity of each change on the sensing traits was +30%, -30%, or zero (no change). Looking for fairness, every time we select a sensory trait to increase, we reduce

Table 1. Observation classes.

Configuration	Modified sensing trait			
	Soccer Twos Game		Tennis Game	
	Max Ray Degrees	Degree Length	Ball Visual	Agent Visual
Ψ_0	100%	100%	100%	100%
Ψ_1	70%	130%	70%	130%
Ψ_2	130%	70%	130%	70%

Table 2. Action classes.

Configuration	Modified actuation trait				
	Soccer Twos Game			Tennis Game	
	Rotation	Lateral Speed	Forward Speed	Speed	Rotation
Φ_0	100%	100%	100%	100%	100%
Φ_1	130%	70%	70%	70%	130%
Φ_2	70%	130%	130%	130%	70%

another one of the same class (see classes Ψ_1 and Ψ_2). We modify in 30% to make a change that will influence and highlight the ability, but we were careful to avoid creating a super agent.

Analogously, we define action classes $\Phi = \{\Phi_0, \Phi_1, \Phi_2\}$, according to some actuation traits that may be modified – see Table 2. In this case, the following actuation traits were selected to be changed in the Soccer Twos game: (i) *'Lateral Speed'*; (ii) *'Forward Speed'*; and (iii) *'Rotation'*. *'Lateral Speed'* and *'Forward Speed'* are related to the velocity that an agent can achieve in the specified direction, while *'Rotation'* indicates the velocity that an agent can rotate around its axis. Likewise, in the Tennis game, the traits were: (i) *'Speed'*; (ii) *'Rotation'*. They represent the speed of the agent in its field and the rotation that the agent ('racket') has, respectively. Table 2 shows the changes in the actuation traits for each action class, in which action class Φ_0 represents the baseline, i.e., no change.

Again, the intensity of each change on the actuation traits was +30%, -30%, or zero (no change), and, once more, every time an actuation trait increases, another one of the same class is reduced (see classes Φ_1 and Φ_2). But in this case, both speeds were treated as one.

5.3. Experimental Settings

Proximal Policy Optimization (PPO) [Schulman et al. 2017] was used as the training method and it is already implemented in the Unity Ml-Agents Toolkit. We did not use the Soft Actor-Critic (SAC) because the adversary is frequently changing, and many situations seem to exhibit non-stationary dynamics from the perspective of a single agent. This could seriously affect the SAC experience replay system [Foerster et al. 2017]. Nevertheless, our proposal is not influenced by the algorithm selection. So any training algorithm can be used with our model for heterogeneous agents.

Table 3 shows the four experiment configurations made in the Soccer Twos game.

Table 3. Soccer Twos experiment configurations.

Configurations	Blue team		Purple team		Blue team		Purple team	
	Agent 1		Agent 2		Agent 3		Agent 4	
	Φ	Ψ	Φ	Ψ	Φ	Ψ	Φ	Ψ
Baseline	Φ_0	Ψ_0	Φ_0	Ψ_0	Φ_0	Ψ_0	Φ_0	Ψ_0
S-Mod	Φ_1	Ψ_0	Φ_1	Ψ_0	Φ_2	Ψ_0	Φ_2	Ψ_0
A-Mod	Φ_0	Ψ_1	Φ_0	Ψ_1	Φ_0	Ψ_2	Φ_0	Ψ_2
SA-Mod	Φ_1	Ψ_0	Φ_2	Ψ_0	Φ_0	Ψ_1	Φ_0	Ψ_2

Table 4. Tennis experiment configurations.

Configurations	Blue Team		Purple Team	
	Φ	Ψ	Φ	Ψ
Baseline	Φ_0	Ψ_0	Φ_0	Ψ_0
S-mod	Φ_0	Ψ_1	Φ_0	Ψ_1
A-mod	Φ_1	Ψ_0	Φ_2	Ψ_0
SA-mod	Φ_1	Ψ_2	Φ_2	Ψ_1

The first one is the baseline in which all agents are homogeneous. The second one, Sensing Modification (*S-Mod*), presents heterogeneous agents in which only observation classes are different, but each team possesses an agent equivalent to the other team. The third experiment, called Actuation Modification (*A-Mod*), represents another scenario with heterogeneous agents, but, in this case, only action classes are different (and, again, each team has an agent equivalent to the other team). Finally, in the last experiment, called Sensing/Actuation Modification (*SA-Mod*), both observation and action classes are distinct for agents belonging to the same team, and no agent from a team is equivalent to any other belonging to the opposite team.

Equivalent to the experiment configuration shown in the Soccer Twos game, we apply the same strategy to the Tennis game. Having four experiments: baseline, Sensing Modification (*S-Mod*), Actuation Modification (*A-Mod*), and Sensing/Actuation Modification (*SA-Mod*), as shown in Table 4.

In the baseline experiment, no change was made in the default values of any sensing nor actuation trait. The other experiments (*S-Mod*, *A-Mod*, and *SA-Mod*) followed the configurations presented in Table 3 and Table 4.

5.4. ELO Rating System

ELO [Düring et al. 2019] rating system is a method that measures the relative skill level between two players. It was created to improve the rating system in chess, and then its usage grew, becoming widely used nowadays. It is used as a rating system in many sports and leagues, such as the National Basketball Association (NBA), National Football League (NFL), Major League Baseball (MLB), and also in games such as Counter-Strike: Global Offensive (CS:GO) and League of Legends (LoL). We used the ELO system as the metric to evaluate the experimental results. The ELO of a player is a number that changes depending on the result of a match. A player who wins takes some points from the loser. If two players with the same ELO play against each other, they should expect the same amount of wins. Thus, ELO rating could be used to rate players. The United States Chess Federation (USCF) suggested scaling ratings so that a difference of 200 rating points in chess would mean that the stronger player has an expected score of approximately 0.75.

5.5. Result Evaluation

Each experiment configuration was run three times and the average of those three runs was used as the experiment's final result to allow a fair comparison among them.

5.5.1. Soccer Twos Game Results

All scenarios with heterogeneous agents achieved a better result than the baseline. That shows that heterogeneous agents have learned to cope better with the test task than homogeneous ones. One possible explanation for that result is that the use of heterogeneous agents allows rarer events to happen. And, with that rare and unique experiences, agents learn more than dealing with the same situations.

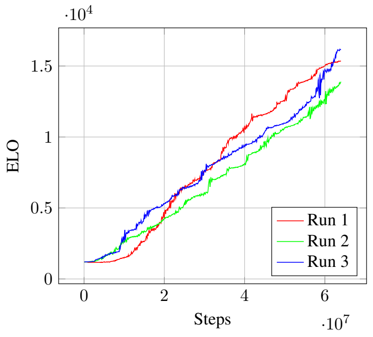
The first experiment is *S-Mod*, in which just the sensing traits were changed. The results of each of the three runs are in Figure 3(a). One can observe that *S-Mod – Run 1* started with the worst result and around 19M iterations it surpasses the *S-Mod – Run 2*. Between 22M and 35M iterations both *S-Mod – Run 1* and *S-Mod – Run 3* got very similar. After that, *S-Mod – Run 1* had a superior result until around 62M iterations, when *S-Mod – Run 3* got to the top again and finishes with the higher score among the three runs of *S-Mod*.

The second experiment is *A-Mod* with changes only in the actuators of each agent. The plot with the results is shown in Figure 3(b). *A-Mod – Run 1* had the lowest result of all the three runs during the entire experiment. *A-Mod – Run 3* touched three times *A-Mod – Run 2*, around 14M, 20M, and 39M iterations. But, besides that, *A-Mod – Run 2* was at the top for almost the duration of the experiment.

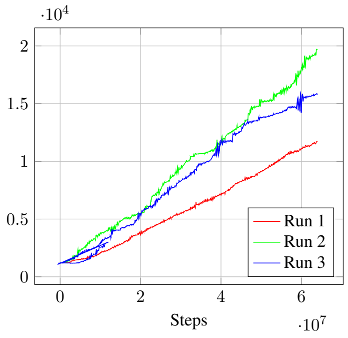
At last, the experiment *SA-Mod* which have both sensing and actuation traits changed has its results shown in Figure 3(c). During the entire experiment, all three results maintain without touching each other. The best run was *SA-Mod – Run 1*, followed by *SA-Mod – Run 2*, and *SA-Mod – Run 3*, at the last position.

Figure 4 presents the evolution of average ELO results of each experiment. It is easy to observe that the results of the three scenarios with heterogeneous agents are better than the baseline (with only homogeneous agents). Since the results for experiments with heterogeneous agents are very close to each other, Table 5 gives a detailed view of those results. The experiment *SA-Mod* presented the best result during almost its entire duration, losing to *A-Mod* in just two points – see values at 8M and 48M iterations in Table 5.

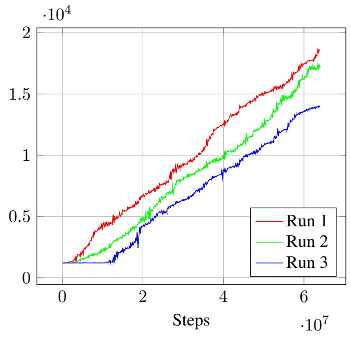
Besides getting better results in ELO rating, *SA-Mod* also presented a particular



(a) ELO results for Soccer *S-Mod*



(b) ELO results for Soccer *A-Mod*



(c) ELO results for Soccer *SA-Mod*

Figure 3. Evolution of ELO results for Soccer Twos, 3 runs of each experiment.

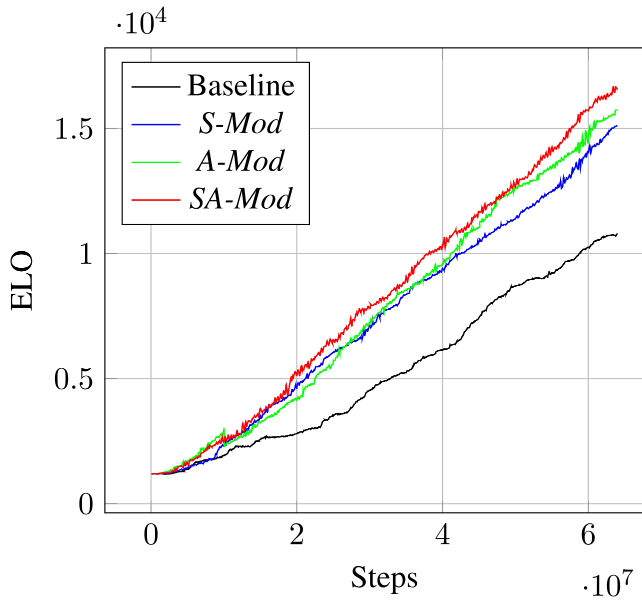


Figure 4. Evolution of average ELO results for Soccer experiments.

behavior. In Figure 5, it is possible to observe that agents learned somehow to organize themselves in *SA-Mod*. One agent belonging to each team went to the attack position, while the other stayed in the defensive field. In the other experiments *S-Mod* and *A-Mod*, it was not possible to observe this behavior. That indicates that agents' heterogeneity allowed them to learn how to position themselves in the game field based on which one possesses the better skill set for each role. That can also explain the best results in the ELO rating presented by *SA-Mod*.

The absolute final result of *SA-Mod* was 16,544 points, and the second-best result is *A-Mod* with 15,730 points. The difference between these two approaches seems to be close in the plot, but for the ELO metric, this gap is huge. If both strategies were put against each other in the same environment, by the ELO calculations the *SA-Mod* will win 99.56% of the matches.

5.5.2. Tennis Game Results

In the first experiment, *S-Mod*, the results of each of the three runs are in Figure 6(a). It is possible to observe that *S-Mod – Run 3* started with the best result and around 12M iterations it starts to go down. The *S-Mod – Run 1* and *S-Mod – Run 2* got very similar

Table 5. Detailed average ELO results for each experiment in Soccer Twos.

	# of Iterations ($\times 10^6$)							
	08	16	24	32	40	48	56	64
Baseline	1,770	2,660	3,356	4,907	6,143	8,281	9,409	10,806
S-Mod	1,796	3,780	5,855	7,729	9,412	11,134	12,717	15,129
A-Mod	**2,393**	3,474	5,343	7,768	9,593	**12,273**	13,724	15,730
SA-Mod	2,261	**3,824**	**6,310**	**8,187**	**10,343**	12,215	**14,404**	**16,544**

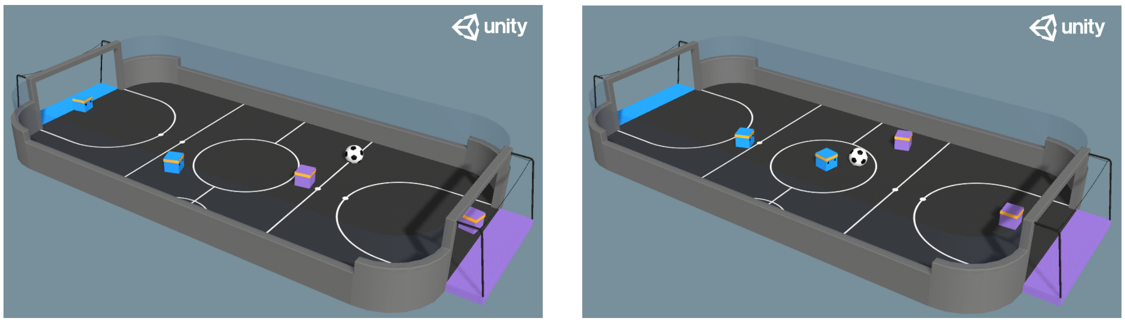


Figure 5. Illustration of Soccer Twos agents organized in offensive and defensive roles in the experiment *SA-Mod* with heterogeneous agents in which sensing and actuation traits were modified.

behavior, crossing results a few times.

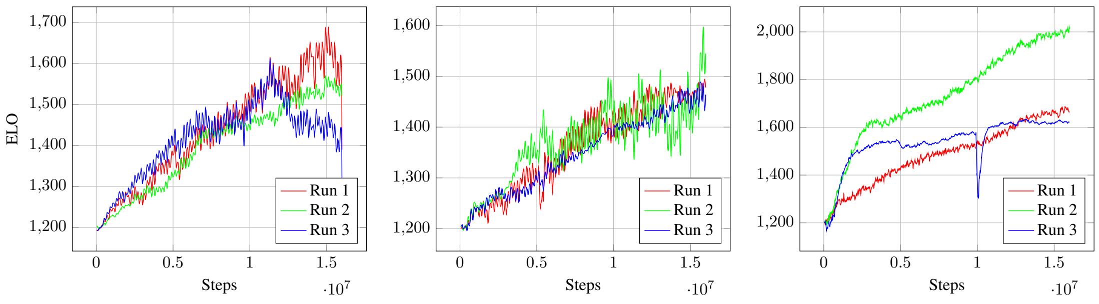
The second experiment is *A-Mod*. Figure 6(b) plots the results of this experiment. All three runs have very similar behavior and values. The *A-Mod – Run 2* has an idle result, but aside from that, it follows the other curves.

At last, the experiment, *SA-Mod*, has its results shown in Figure 6(c). *SA-Mod – Run 1* has an almost continuous growth, finishing the experiment better than the *SA-Mod – Run 3*. The *SA-Mod – Run 3* had a fast growth at the beginning of the test but after 2M iterations, the growth decreases. In around 10M iterations it suffers an attenuated loss but recovered the result in less than 1M iterations later. The *SA-Mod – Run 2* accomplished the best result of this experiment, followed by *SA-Mod – Run 1*, and *SA-Mod – Run 3*, at the last position.

Figure 7 shows the average ELO results of the three runs. It is possible to observe that the result of *SA-Mod* presented the best result of all experiments. In Table 6 is possible to see the detailed view of the results. The *A-Mod* and *baseline* (with only homogeneous agents), had a very similar result, apart from that, the *S-Mod* had a slightly better result.

The final absolute result of *SA-Mod* was 1766 points, followed by *S-Mod* with 1,515 points. For the ELO metric, if an agent of the *SA-Mod* were put against the *S-Mod* agent, by the ELO calculations the *SA-Mod* agent will win 81% of the time.

An issue related to making agents heterogeneous in competitive-only games (with single-agent teams) by increasing one agent's skill and another skill of its opponent is



(a) ELO results for Tennis *S-Mod* (b) ELO results for Tennis *A-Mod* (c) ELO results for Tennis *SA-Mod*

Figure 6. Evolution of ELO results for Tennis, 3 runs of each experiment.

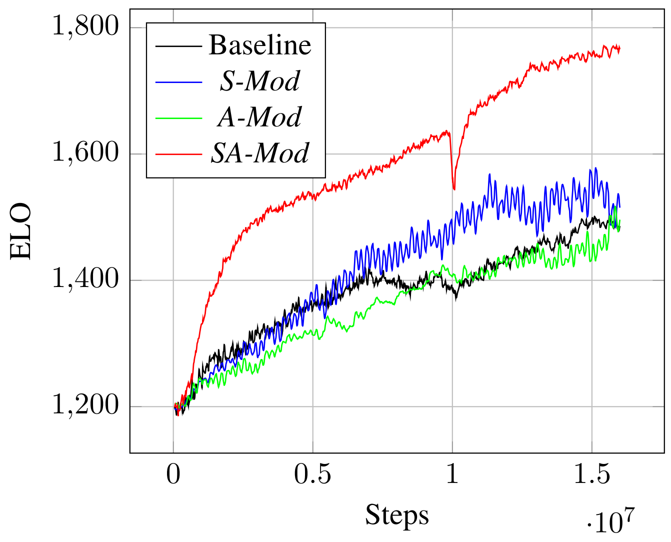


Figure 7. Evolution of average ELO results for Tennis experiments.

the possibility of not being fair. For example, which skill is more critical in the game of Tennis: the observation of where the ball is or the force used to hit the ball? This question is not easy to answer. Furthermore, it's possible that instead of training heterogeneous agents fairly, we will just discover the best skill set needed for a given game.

Finally, one should note that a downside of the proposed approach is that homogeneous agents generally allow for parameter sharing and simpler network architectures, which can lead to faster training [Wakilpoor et al. 2020].

6. Conclusion and Future Works

In this work, we propose a model for heterogeneous agents that can observe the environment with distinct acuity levels and also act on it with different intensity magnitudes. All scenarios with heterogeneous agents achieved a better result than the baseline (with only homogeneous agents). Best results are achieved when agents are more diverse, i.e., agents belonging to the same team with distinct observation and action classes and not equivalent to any other belonging to the opposite team. These results were obtained in both environments that we tested, a cooperative and competitive game and a competitive-only game.

In future works, it is possible to investigate different changes in sensing and actuation traits when defining observation and action classes. It is also interesting to apply the proposed model to cooperation-only games to assess its results.

Table 6. Detailed average ELO results for each experiment in Tennis.

	# of Iterations ($\times 10^6$)							
	02	04	06	08	10	12	14	16
Baseline	1,276	1,353	1,370	1,389	1,385	1,434	1,459	1,486
S-mod	1,274	1,338	1,390	1,437	1,478	1,512	1,541	1,515
A-mod	1,257	1,302	1,329	1,371	1,406	1,430	1,426	1,496
SA-mod	**1,437**	**1,525**	**1,549**	**1,589**	**1,562**	**1,707**	**1,750**	**1,766**

Acknowledgements

This study was financed in part by PUC Minas and by the Fundação de Amparo à Pesquisa do Estado de Minas Gerais — FAPEMIG.

References

Bowling, M. and Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172.

Düring, B., Torregrossa, M., and Wolfram, M.-T. (2019). Boltzmann and fokker–planck equations modelling the elo rating system with learning effects. *Journal of Nonlinear Science*, 29(3):1095–1128.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 1146–1155. PMLR.

Fu, Q., Ai, X., Yi, J., Qiu, T., Yuan, W., and Pu, Z. (2022). Learning heterogeneous agent cooperation via multiagent league training. *arXiv preprint arXiv:2211.11616*.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., and Lange, D. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Li, Y., Wang, X., Wang, J., Wang, W., Luo, X., and Xie, S. (2020). Cooperative multiagent reinforcement learning with hierarchical relation graph under partial observability. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1–8. IEEE.

Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., and Graepel, T. (2019). Emergent coordination through competition. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Lowe, R., WU, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M. (2018). Reinforcement learning for solving the vehicle routing problem. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Oroojlooy, A. and Hajinezhad, D. (2023). A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, 84(1-2):109–136.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Vlassis, N. (2022). *A concise introduction to multiagent systems and distributed artificial intelligence*. Springer Nature.

Wakilpoor, C., Martin, P. J., Rebhuhn, C., and Vu, A. (2020). Heterogeneous multi-agent reinforcement learning for unknown environment mapping. *arXiv preprint arXiv:2010.02663*.

Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.