Multi-agent Meta Reinforcement Learning via Hierarchical Interactions

Orazio Rillo
Ecole Polytechnique Fédérale de Lausanne

***Abstract*—Reinforcement learning has proven to be very successful in solving various learning problems, that go from the domain of games to that of robotics. However, the dominant strategy in many practical applications consists in adapting the model to the specific task to be performed. This project aims to introduce a framework in which agents are able to adapt by themselves to the task. While trying to understand which is the most effective way of doing it, we will explore new possibilities in multi-agent reinforcement learning, in order to find a configuration that can enable an easy adaptation of our model to any task that needs to be solved. By taking a cue from the meta-learning philosophy, we want to train our agents not to perform the task, but to learn how to learn to perform the task. Ultimately, our approach will consist in combining hierarchical reinforcement learning and meta-learning, by meta-training an agent (the 'manager') to learn how to communicate a reward function in such a way that the (potentially many) other agents (the 'workers') are guided by the former to learn how to execute a task.**

I. Introduction

Traditional reinforcement learning is not an option in multi-agent environments. It scales poorly with problem size, since the state space grows exponentially in the number of agents. Therefore, in recent years various research works trying to address this issue have arisen. In particular, in 2016 Kulkarni et al. introduced a new framework specifically built to solve goal-directed tasks in complex environments, called Hierarchical-DQN [7]. More generally, hierarchical reinforcement learning (HRL) "divides the overall task into independent goals and trains a meta-controller to pick the next goal, while a controller learns to reach individual goals" ([8]). Since then, other researchers were inspired by the idea of having a hierarchical system among the agents, and frameworks such as Feudal Multi-Agent Hierarchies [2] have arisen. We are part of those researchers. Indeed, this project was driven by the idea of creating a new framework in which a group of agents could manage to cooperate and learn how to solve a task, regardless of all the specifications of the environment.

II. Related Works

This project takes inspiration from the work of Ahilan & Dayan already cited in the introduction. With the idea of investigating how agents can learn to cooperate, they introduced a new HRL framework and called it Feudal Multi-Agent Hierarchies (FMH). It can be summarised as follows: a 'manager' has the task of maximising the environmental reward; in order to do this, it has to learn to communicate subgoals to other agents, called 'workers', that simultaneously work together to achieve the subgoals communicated by the manager.

We integrated the idea of FMH with that of meta-learning, which is a set of machine learning algorithms that learn from the output of other machine learning algorithms. Meta-learning has already been employed in different research works with a reasonable success because "it aims to learn a common structure between different tasks so as to solve new and relevant tasks more efficiently" ([6]). The main references behind our use of meta-learning come from the papers by Zou et al., 2019 [15] and Bechtel et al., 2019 [3].

For the implementation of reinforcement learning algorithms, we used RLlib [9], an open-source library for reinforcement learning that offers both high scalability and a unified API for a variety of applications.

Finally, in all our experiments we used an environment inspired by Lowe et al. [11], called Multi-Agent Particle Environment. More specifically, we reproduced the environment referred in the paper above as "Cooperative Communication", then we

slightly modified it to make it compatible with RLlib and to adapt it to the experiments we wanted to perform.

III. Methods

We proceeded by increasingly complex steps. We started by creating and refining the environment in which our agents had to live. Then, we attempted to reproduce the results of the paper by Ahilan & Dayan [2]. Besides, we made some experiments that go beyond the scope of the mentioned paper, with the idea of comparing how much the agents could benefit from different observation and action spaces. Finally, we implemented a new framework that combines FMH with some meta-learning ideas, in which we meta-trained the manager to make it learn how to effectively communicate a reward function to the worker, that in turn was the agent in charge of actually solving the task.

A. Base Environment

As already mentioned, the environment we used in our experiments is an adaptation of the Multi-Agent Particle Environment from OpenAI. It consists in a space containing a certain number of coloured landmarks, where two cooperative agents, one called 'speaker' and the other called 'listener', are placed. The objective is to steer the listener to a target landmark, randomly selected at each episode. The speaker is unable to move and to observe the listener. Still, it observes the color of the goal landmark and can communicate with the listener at each time step. On the contrary, the listener is able to move, but cannot communicate with the speaker. It can observe: its velocity, its relative position with respect to each landmark and the speaker's communication, but it is not aware of the colour of the landmark it must navigate to. The environmental reward is given proportionally to the negative distance of the listener to the goal landmark. However, note that while the speaker obtains its reward directly from the environment, the listener computes it locally, proportionally to the inverse distance that separates it from the communicated goal. The reason behind this is that we want the listener to follow the speaker's communication regardless of the ability of the manager to correctly guide it. A successful execution consists in the speaker correctly navigating the listener to the target landmark.

There are two remarks that we think it is important to make:

1) though very simple, this environment intrinsically has an extreme variability, since the coordinates and the color of each landmark is randomly selected at the beginning of each episode;
2) what we described is what we called the "base environment", then it is slightly modified according to the constraints imposed by the specific experiment to run.

B. Single-Agent

The first method that we tested is single-agent learning. It is clear that the environment that we described above was designed to work by exploiting agents' cooperation. However, in this experiment we just wanted to test how an agent that receives a sparse reward performs in such a task. Thus, we modified the base environment by removing the manager agent and modifying the way rewards are distributed. More specifications about the modification we made can be found in the next section.

C. Feudal Multi-Agent Hierarchies

We implemented FMH, the framework proposed by Ahilan & Dayan in 2019. They addressed three major issues in multi-agent reinforcement learning (i.e. non-stationarity, scalability and coordination, even though we did not address the problem of scalability in this project) with a simple yet powerful idea: defining a hierarchical structure among the agents. Such a structure can be imagined as a rooted directed tree, in which the highest level manager corresponds to the root node and each worker acts in order to achieve the subgoal communicated by a single manager. A correct assignment of the roles of manager and worker is crucial in order to obtain good results. In the environment described in section III-A, it naturally comes to assign the role of manager to the speaker and the role of worker to the listener.

As discussed in [2], we addressed the problem of non-stationarity in two ways. The first was to introduce a 'pre-training' procedure, in which the worker was trained before training the manager.

Note that, even though the manager is not trained in this procedure, it still acts in the environment. The idea behind pre-training is that of teaching the workers to achieve managerial subgoals before the actual training procedure starts. About this, Ahilan & Dayan write: "As subgoals are initially of (approximately) equal value, the manager will explore them uniformly. If the set of possible subgoals is sufficiently compact, this will enable workers to gain experience of each potential subgoal". The second way non-stationarity was addressed was to introduce a heuristic that Ahilan & Dayan called "temporally extended subgoals", to reduce the noise of speaker's communications. It consists in "forcing" the manager to commit to a subgoal for an extended period of time, ideally long enough to allow the worker to, at least, partially achieve it.

The other problem that we mentioned is coordination. A good coordination is essential in achieving a good performance in a cooperative task; therefore, it is in manager's own interest to learn how to effectively communicate with the workers and to coordinate them in order to maximise its reward.

Finally, let us define our notion of convergence. Similarly as in [2], it was determined by comparing the mean reward obtained by the manager in one epoch with the mean reward computed within a sliding window 5 epochs large. At the end of each epoch, if the absolute difference of these two values is smaller or equal to the 2% of the mean reward in the sliding window, we increment a counter. If and only if this counter is incremented 5 times in a row before the last 10 epochs of the training, we have convergence.

D. Meta-learning FMH

We built meta-learning upon the baseline structure given by FMH.

In order to do it, we used the environment variant in which the manager does not communicate the goal of the workers at each time step, but the reward function directly. We will talk about this modification in more details in the next section.

With this method, our idea was that of teaching to the manager how to effectively communicate a reward function in order to navigate the workers to their targets. In order to do that, we built a framework in which the manager and the worker are trained separately. At each meta-training iteration, we tried to improve the performance of the manager by searching a better set of parameters. The evaluation of these sets is performed by executing a training procedure in which only the worker network is updated. Still, the manager participates by exploiting its knowledge and communicating the reward function to the worker according to its observations. The "inner" training loop (i.e. the one that only involves the worker) is based on the same algorithm we used to train our agents in vanilla FMH. On the contrary, the "outer" meta-training loop exploits non-gradient-based methods. Note that the evaluation of a set of manager weights has to take into account not only the final reward that the manager achieves, but also the speed with which the algorithm converges, therefore the mean reward is not a good metric. We will discuss the metric we picked to address this problem and all the other details about this framework in the following section.

IV. Experiments and Results

In all the experiments that follow, we have exactly two agents, a manager and a listener (with the exception of the single-agent case), a total of 12 landmarks on the map, each of which is characterised by a different color. In order to clearly distinguish them, we used the same color palette employed in [2], i.e.

[0.65, 0.15, 0.15], [0.15, 0.65, 0.15], [0.15, 0.15, 0.65], [0.65, 0.65, 0.15], [0.15, 0.65, 0.65], [0.65, 0.15, 0.65], [0.65, 0.65, 0.65], [0.15, 0.15, 0.15], [0.40, 0.40, 0.65], [0.40, 0.40, 0.15], [0.15, 0.40, 0.40], [0.65, 0.40, 0.40].

A. Single-agent

Our first experiment sees as main (and only) actor the worker, which is not allowed to cooperate with the manager in order to achieve the goal. We decided to use a very common pattern in single-agent reinforcement learning, i.e. the worker, instead of locally computing its own reward based on its observations, receives a sparse reward whenever it reaches a goal. Specifically, the algorithm we used to train the agent is Proximal Policy Optimisation (PPO) [13]. The networks that implement the behaviour of the agent are two fully connected feedforward networks with two hidden layers with

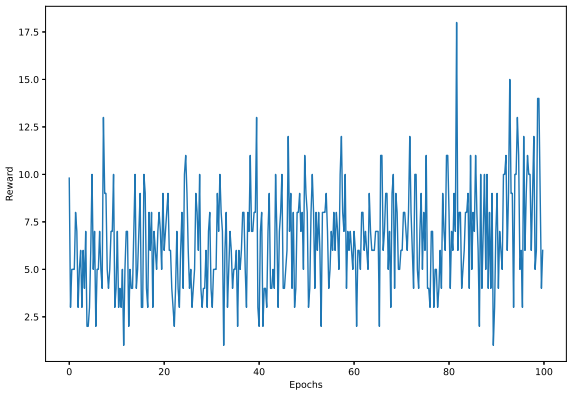

Fig. 1. Single-agent training - mean worker reward on the y-axis

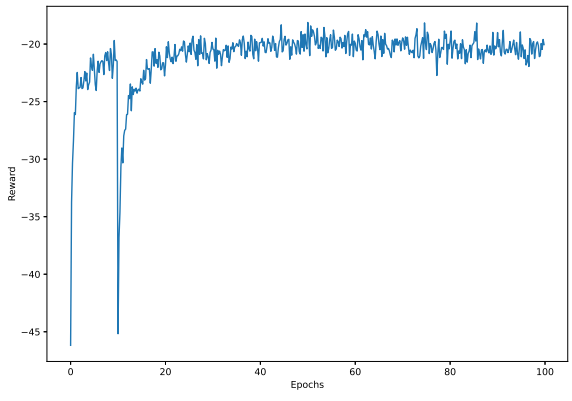

Fig. 2. Standard FMH training - mean manager reward on the y-axis

256 neurons per layer. As optimiser, we picked Adam, with a learning rate of 0.001 for both the policy and the value network. The discount factor is $\gamma = 0.75$. We trained the agent for 10 epochs; each of them is composed of 1000 episodes, which in turn are limited to 25 steps. The reward given by the environment when the agent reaches the target is 100, in any other scenario no reward is distributed.

In Figure 1, we can see the behaviour of the agent during the training. It is clear that it does not learn how to reach its target, but most likely it moves following a random policy and occasionally it reaches its objective. Indeed, from the plot we can see that it gets an average reward of nearly 8 in the last episodes of the training, which means that the target is hit nearly 8 times out of 100. Note that we did not expect such an algorithm to work. In fact, as we previously mentioned, the position of the particles changes at each episode. In addition, recall that the worker is not aware of the position of the target landmark. For these two reasons, we think that the poor performance of the agent is justified by the excessive difficulty of the task.

B. FMH-PPO

Our second experiment consisted in attempting to reproduce the results obtained in [2]. Therefore, we implemented FMH as described in the relative paper. The only difference that exists between our implementation and that of Ahilan & Dayan is the algorithm that has been internally used to train both the speaker and the listener, which is PPO [13] in our case and DDPG [10] in their experiments. We preferred PPO to DDPG because it allowed us to have a much faster training and to use discrete spaces for the communication of the goal in RLlib implementation of algorithms. We used a "temporally extended subgoals" heuristic to enforce an extended communication of manager 8 time steps long. The training consisted in a pre-training of 10 epochs and an actual training for the remaining 90 epochs. Again, each epoch is made of 1000 episodes.

In Figure 2, it is self-evident how the agents learn how to coordinate in order to accomplish the goal[1]. In addition, we think it is relevant to remark that the algorithm converges very fast. However, it is difficult to compare our results to those in the paper by Ahilan & Dayan. We can clearly see a positive training trend, but we cannot trust our attempt to be a perfect replication of the original framework, since the paper misses some implementation details and key information to correctly interpret the plots. Nonetheless, our concern was not to achieve the best performance possible, nor to outperform FMH, but

[1]The drop that we can observe in Figures 2 and 3 at epoch 10 is due to some RLlib implementation detail. In fact, at epoch 10, we save a checkpoint and then we interrupt the pre-training procedure. After that, we restore the trainer with a new configuration that will be used in the actual training loop. to However, every time a trainer is restored from a checkpoint, it takes a few epochs before setting the old performance in order again.

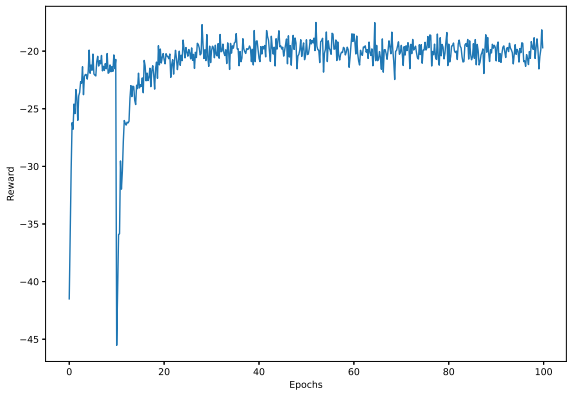

Fig. 3. Variant of FMH discussed in subsection IV-C1 - mean manager reward on the y-axis

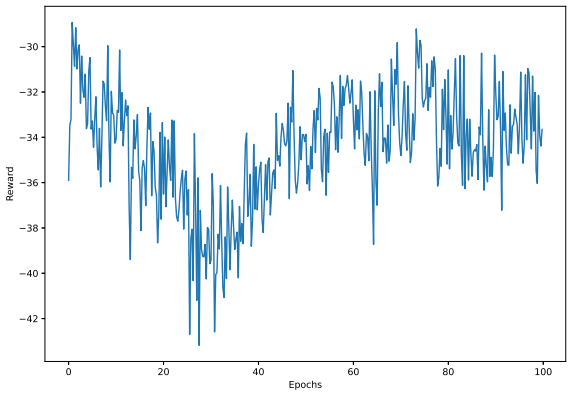

Fig. 4. Variant of FMH discussed in subsection IV-C2 - mean manager reward on the y-axis

to get a baseline for the comparison with the other simulations we wanted to test.

C. Variants of FMH

We explored a few variants of the standard FMH.

1) Changing the worker reward: the first variant is very similar to standard FMH-PPO. The only difference is that the worker receives as reward the average of the locally computed reward (given by the inverse distance from the communicated goal) and the environmental reward (which corresponds to the same reward obtained by the manager i.e. the inverse distance of the worker from the real target). Performance-wise, we observe no fundamental difference with FMH, as we can see in Figure 3.

2) Communicating the reward function: the second variant we analysed is motivated by the idea of moving some steps towards the meta-training approach. Therefore, we re-imagined the environment by changing the way the speaker communicates to the listener. In fact, while the reward of the manager is given by the environment (similarly to FMH), the reward of the worker is not computed locally anymore. On the contrary, it is directly communicated by the manager. So, manager's communication is not a positive discrete value in the range of the number of landmarks anymore, but a continuous value in the range (-50, 50). Therefore, the speaker needs to learn how to effectively communicate this reward function, in order to navigate the listener to its goal. Of course, in such a scenario it makes no sense to use any pre-training, nor any extended communication heuristic. Figure 4 shows a much noisier training with respect to the previous simulations. In addition, the performance gets worse in the first epochs of the training and then slightly improves in the successive ones. Overall, the agents do not seem to actually improve their coordination in time. Therefore, it will be interesting to see whether combining this approach with meta-learning will lead to interesting results.

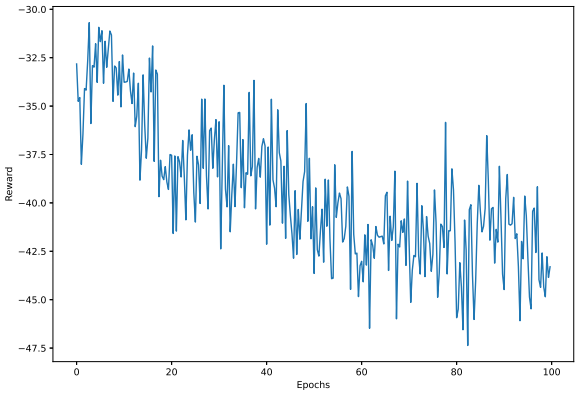

Fig. 5. Variant of FMH discussed in subsection IV-C3 - mean manager reward on the y-axis

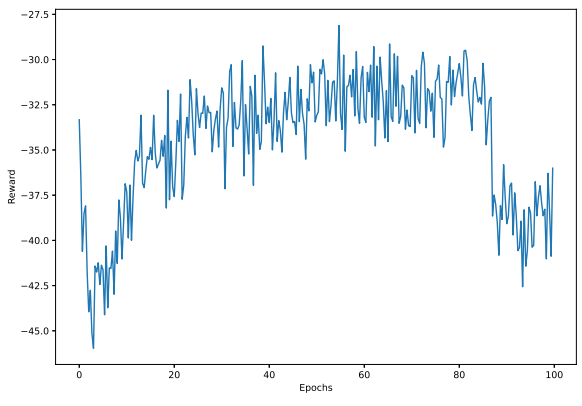

Fig. 6. Variant of FMH discussed in subsection IV-C4 - mean manager reward on the y-axis

3) Communicating the reward function and allowing the manager to observe the relative position of the worker from the goal: the third variant is quite similar to the second one; the only difference between the two is what the manager observes. Indeed, we tried to give more information to the speaker, hoping that this would have led to a performance improvement. So, the manager observes not only the goal of the worker, but also the inverse distance of the worker from the goal. However, this modification did not lead to any relevant result. Looking at Figure 5, the overall performance seems to be even worse than that achieved in the previous simulation, though it is very likely that randomness plays a relevant role in this simulation; in both cases the training is so noisy that we cannot really compare them with enough confidence.

4) Communicating the reward function and allowing the manager to observe the relative position of the worker from each landmark: the fourth and last variant of FMH we tested is again a tentative of improving the performance that the agents achieved in subsection IV-C2. Therefore, the overall structure is the same, but we tried to increase the amount of information that the speaker has access to even more, with the same idea we discussed in the previous point. In this experiment, the manager can observe both the goal of the worker and the inverse distance of the worker from each landmark. The mean reward obtained during the training is showed in Figure 6. The plot shows some improvement of the performance during the first 80% of the training procedure, however this is not constant, nor consistent. Thus, the overall performance does not end up being particularly significant.

D. Meta-learning FMH

As our experimental results showed so far, training the speaker to effectively communicate a reward function seems to be a very hard task. Therefore, believing that the problem was that we were trying to train the manager with a gradient-based method, we decided to use a new framework, based on the notion of meta-learning. In fact, the speaker is not trained in the traditional way, but exploring the space of the possible combinations of parameters of the manager networks. In all the other experiments, the neural networks have two hidden layers each and 256 neurons per layer. It is clear that this is a space too big to be explored by any algorithm, thus we reduced the complexity of the network, though maintaining the other parameters unchanged. A key factor for a meaningful training with such a framework is to decide on what metric(s) to evaluate the sets of weights, in order to distinguish those that we want to discard from those that we want to keep. As already mentioned in III-D, the selected metric had to take into account both the speed of the training and the overall performance achieved. Therefore, we opted for the total accumulated reward during the evaluation. Indeed, the evaluation of a set of parameters consists in initialising a manager network with the parameters that we want to test, then performing a standard training procedure in which only the worker network is updated. Apart from this modification, the other hyperparameters are unchanged compared to the baseline structure, i.e. the FMH variant described in section IV-C2. We run two simulations with the new framework, each of them using a different meta-training algorithm.

1) Random Search (RS): this algorithm is quite commonly used for hyperparameter optimisation [4]. In this simulation, we tried to use it to meta-train the manager network. We started from an arbitrary initial configuration of the parameters (we use RLlib's default initialisation). Then, we converted these parameters into their binary representation. The space to explore is therefore be $p \times n$-

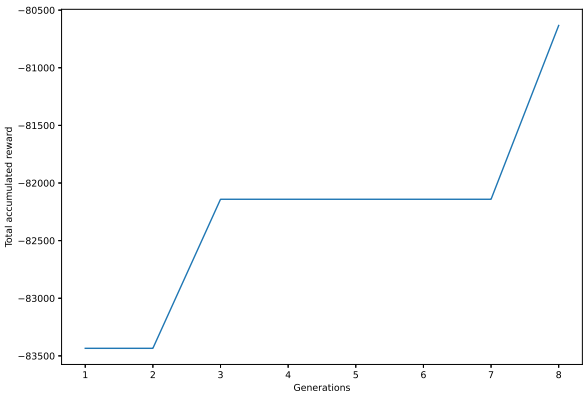

Fig. 7. Meta-learning FMH with GA - total accumulated reward on the y-axis

dimensional, where p is the bit-precision of the binary representation of the parameters and n is the number of parameters to optimise. After the initialisation, we sample a vector onto the unit hypersphere built around the initial point. If the sampled vector (converted back to a representation suitable for the training) achieves better performance than the reference one[2], the new vector becomes our reference and the procedure is repeated. We reduced the hidden layers' size of both the manager and the worker networks to 16 to shrink the space to explore. We trained our model for 100 epochs, with 5000 episodes each. In every epoch, we evaluate our current set of parameters and then we perform the random search update.

The training took nearly 16 hours and the total accumulated reward in the best epoch has been -134094.3 for 5000 episodes, which corresponds to an average reward of -26.81. This does not beat FMH, but it is already a nice improvement compared to the variant that we used as baseline. However, we acknowledge that random search is not a very powerful algorithm, especially in scenarios in which the space to explore is as wide as in our case. In addition, it suffers from getting stuck in local minima. This motivated our choice to test the same framework with the following algorithm.

[2]The reference vector is initialised as equal to the initial configuration of the parameters.

2) Genetic Algorithm (GA): this is a very popular algorithm for solving optimisation problems based on a natural selection process that mimics biological evolution [12]. We will not go into the details of the implementation of our algorithm, since we used a pretty standard version of GA. We opted for using it since it is noticeably more powerful than random search in exploring wide domains. Our pipeline consisted in initialising the network with RLlib's default set of parameters, then we converted each of them into its binary representations and finally we recombined the binary vectors by using GA to find the best set possible. Our network is made of two hidden layers, with 64 neurons per layer. This choice was motivated by the needs of having sufficient expressivity to be able to perform well in the task and of not having a too large network at the same time. We meta-trained the speaker by running a genetic algorithm over a population 400 individuals for 8 generations and the computation of the "score" (evaluation metric of the GA, analogous to that of RS) is done by training the worker agent for 3000 episodes, while the manager is never updated.

As shown in Figure 7, the total accumulated reward of the best individual was discovered in the 8th generation and is -80632.9, which corresponds to an average of -26.88. Performance-wise, the same considerations we did for RS hold. However, here we think it is important to remark that genetic algorithms do not suffer from getting stuck in local minima. This means that, with the hypothesis of having infinite time, we would very likely be able to eventually find the optimal combination of parameters. Even reasoning more pragmatically, the more time we meta-train the speaker, the better our results will be. Therefore, we claim that the potential of this method is steadily higher than that of meta-learning with RS. This is not self-evident by looking at the numerical results because, due to time constraints of the project and to the high computation costs, we were able to train the agents for "only" 8 generations. Still, this training procedure took 8 days to be completed. Nevertheless, this was enough to fulfill the purpose of the project, i.e. to show that our approach is not only possible in theory, but actually applicable in practice.

V. Discussion

We introduced a framework that combines HRL and meta-learning to enable an agent to teach its partner how to achieve a goal. As mentioned, we did not expect this method to outperform FMH, since, in the former, the task to learn is steadily more complicated than the one FMH was trained for.

However, an observation that has not been made yet is the following: in the same environmental conditions, the method discussed in section IV-C2 is not effective in training the manager to correctly communicate the reward function, while meta-learning FMH seems to be much more promising. It reaches a better overall performance and still has the potential to improve by extending the training time. The explanation we gave to this phenomenon is that, in the former framework, the credit assignment of the correct actions in iteration n cannot be made because its effects would only be observed after the gradient update, in iteration $n + 1$. This is another (more subtle) reason why the evaluation of the performance should be done over the whole training to get reasonable results.

Ultimately, non-gradient-based methods are generically slower than gradient-based ones. However, they are an extremely valid alternative when the environmental conditions impede the exploitation of gradient descent. Even though our framework can certainly be improved, we think that it represents a remarkable step towards an universal model, able to exploit its ability of 'learning to learn' to quickly adapt its behaviour to each newly assigned task.

References

[1] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Sanjeevan Ahilan and Peter Dayan. *Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning*. 2019. arXiv: 1901.08492 [cs.MA].

[3] Sarah Bechtle et al. *Meta-Learning via Learned Loss*. 2021. arXiv: 1906.05374 [cs.LG].

[4] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.10 (2012), pp. 281–305. URL: http://jmlr.org/papers/v13/bergstra12a.html.

[5] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[6] Yun Hua et al. *HMRL: Hyper-Meta Learning for Sparse Reward Reinforcement Learning Problem*. 2021. arXiv: 2002.04238 [cs.LG].

[7] Tejas D. Kulkarni et al. *Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation*. 2016. arXiv: 1604.06057 [cs.LG].

[8] Saurabh Kumar et al. *Federated Control with Hierarchical Multi-Agent Deep Reinforcement Learning*. 2017. arXiv: 1712.08266 [cs.AI].

[9] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. 2018. arXiv: 1712.09381 [cs.AI].

[10] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[11] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Neural Information Processing Systems (NIPS)* (2017).

[12] K.F. Man, K.S. Tang, and S. Kwong. "Genetic algorithms: concepts and applications [in engineering design]". In: *IEEE Transactions on Industrial Electronics* 43.5 (1996), pp. 519–534. DOI: 10.1109/41.538609.

[13] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[15] Haosheng Zou et al. *Reward Shaping via Meta-Learning*. 2019. arXiv: 1901.09330 [cs.LG].

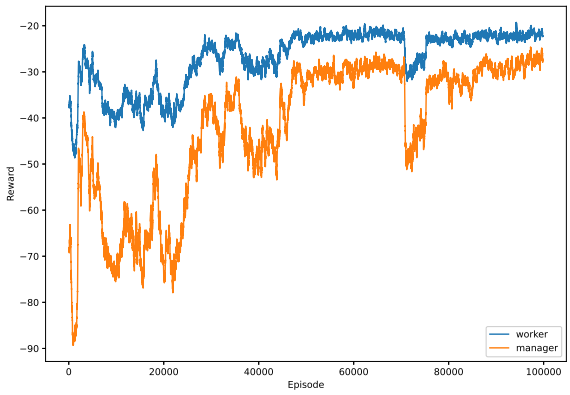

Fig. 8. Standard FMH in which the agents have been trained using REINFORCE

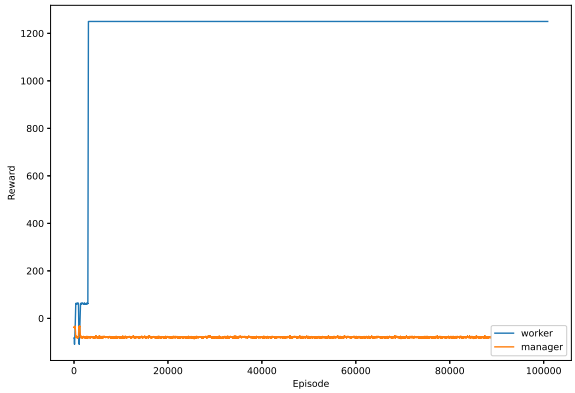

Fig. 9. Variant 2 of FMH in which the agents have been trained using REINFORCE

APPENDIX A
FMH-REINFORCE

To validate the results we obtained by using PPO in some of our learning frameworks we made a couple of experiments implementing by ourselves the policy-gradient algorithm that R. Sutton & A. Barto called REINFORCE in [14]. We implemented it by using the API of Keras [5], which is is a deep learning framework built on top of TensorFlow [1]. In particular, the experiments that we reproduced are those that use the standard FMH and the variant discussed in section IV-C2. The hyperparameters of the training are unchanged in both the versions. As expected, the outcomes of the two training procedures, showed in Figures 8 and 9, validate our previous results. Indeed, in the standard FMH framework, the agents are again able to effectively learn how to coordinate to perform the task, while this does not happen for the variant in which the speaker has to communicate the reward function to the listener. The training of the manager is, in the second case, as noisy as we noticed in subsection IV-C2, however this is not evident from the picture because of the huge scale of the y-axis. Indeed, it is curious to observe how the listener almost immediately learns how to get the highest reward possible from the manager i.e. behaving accordingly to the reward function communicated by the manager, but its partner is not able to navigate him to the correct goal.