PIR : Mixed-initiative mission

(Final report)

Student1*, Student2*, Advisor1†‡ and Advisor2‡
*Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse, FRANCE
Email: {student1,student2}@student.isae-supaero.fr
†Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse, FRANCE
Email: advisor1@isae-supaero.fr
‡Direction Générale de l'Armement - Maîtrise de l'Information, 35998 Rennes Armées, FRANCE
Email: {advisor1,advisor2}@intradef.gouv.fr

***Abstract*—Mixed-Initiative mission are prone to be more and more present, as the use of machines and their autonomy increase rapidly. But this machines always possess a human operator to monitor them, that is thought as omniscient. Our point of view is to demonstrate that the human part is not so reliable, by monitoring the operator and adding those data in the policy of the machine. In this report we develop a reinforcement learning environment for a search and fight mission, test it and show the result for a TD(λ) agent.**

I. Context

The increasing autonomy of robots led to make use of them in a more and more large range of operations and activities, for instance in rough places for surveillance or inspection of contaminated sites. Because the machines cannot be held responsible in case of failure and do not possess the human creativity to improvise, those activities require the assistance of humans. In a mission led by an autonomous robot under the surveillance of a human operator or pilot, the latter is often viewed as a providential agent, capable of resolving any problem and responding to a failure of the robot instantly. But many factors can also lead to a failure of the mission, regarding the human intervention, such as a misinterpretation of the information given to the pilot (due to e.g. a poor designed user interface) or the tunneling of the human operator on a certain task or point of the mission [2]

Regarding the state of the art, the "Fitts list" [1], which is a compilation of the general strengths and weaknesses of machines and humans, was traditionally used to determine the function allocation between human and machines. The allocation function is the function that defines the actions the human and the machines will do and their role during the mission. In order to determine the level of human versus autonomy involvement, it is important to know the individual strengths of each actor. On the one hand, humans are capable of complex social interactions, moral judgment, operate well in dynamics environments and are highly flexible. On the other hand, the machines are capable of higher computational performance, produce repeatable results, can handle multiple tasks simultaneously and are not prone to fatigue or boredom [8].

A problem still lies in the fact that the operator is considered as an external help, capable of resolving all the problems encountered by the machine. In fact, the human operator should no be considered like this, but as a part of a team. But why the human operator should be a part of a team? Because of the previously mentionned issues : the human operator is not fiable. He is subject to boredom, fatigue, stress and many others factors that can contribute to lessen its efficiency during the mission. Considering this fact, the operator should be monitored to consider if he is capable of making accurate decisions, and allow the use of countermeasures (visual stimuli or alarm).

Our idea is to use Probabilistic Planning in order to decide when the system produce countermeasures for the human-operator, and distribute the tasks between human and robot.

While we know all the information about the machine and its states, we know nothing about the states of the human operator. [4] has shown that the use of a Mixed Observability Markov Decision Process (MOMDP) [5] model was relevant to compute a useful policy taking into account the partially observable cognitive state of the operator. MOMDP are a planning framework for computing optimal sequence of actions over time. The sequence is called policy.

The cognitive ability of the human operator can be inferred via an eye-tracker device [7], which shows areas where the operator focuses its attention. Also, the operator's heart beat states, observable via an electrocardiogram (ECG), are successful indicators of cognitive load [9]. The ECG can be coupled with an Eye-Tracker (ET) in order to detect if the operator is in a state of tunneling, and able to take accurate decisions . These data define the observations of the MOMDP.

As mentionned previously, our assumption is that the human operator is not reliable. Our goal is to optimize the mission of a human-robot team using observations from this system. As a proof of concept, we defined a mission of type search and fight, which consists in extinguishing fire that will spread over trees in a area with the help of a robot and a user interface.

The number of fire, the battery level or water level are stressing elements. The navigation is a shared task. The ultimate goal of the mission is the optimization of the number of extinguish fire. The MOMDP model will be built using data from a website (i.e. the user interface), and based on Reinforcement Learning. The work developed in this report concerns the optimization of robot's actions given a human

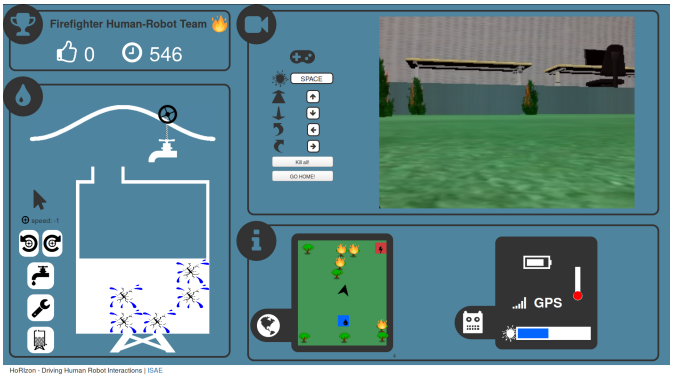

Fig. 1: View of the user interface

behavior. We proposed to compute accurate strategies for the robot by using Reinforcement Learning (RL).

II. Theory

Firstly, a little bit of history for RL : the birth of RL came from the encounter in the late 70s between the computational neurosciences (reinforcement of the synaptic weights of neuronal transmission) and the experimental psychology (animal conditioning models : reinforcement of the behavior leading to satisfaction, *e.g.* Pavlov's dogs experiment) with an adequate mathematical environment : dynamic programming.

A link with experimental psychology is the law of effects of Thorndike (1911) [3] : "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur : those which accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond".

And so, what is Reinforcement Learning? It is an approach, here computational, to decision-making, understanding and goal-directed learning. The individual learns from direct contact with the environment. It only needs the definition of the interactions between a learning agent and its environment, in terms of states, actions and rewards. The states represent the description of the environment. The actions are the possible agent's actions. The rewards define the goal.

The value and value functions are concepts that are key features of RL methods. Value functions are essential for an efficient search in the space of policies. The value function are defined over the state space :
for $x \in X, V(x)$ is the mean of the sum of the rewards over time for a process starting in x.

$$V^{\pi}(x) = \mathbb{E}[\sum_{t \geqslant 0} r(x_t, \pi(x_t))]$$

Continuing on that point, what are the objectives of RL? The objectives of RL are : the automatized acquisition of skills for decision making in a complex and uncertain environment and learning by "experience" a behavioral strategy (named policy) reflecting the failures and success (reinforcements or rewards).

Some examples of achievements of Reinforcement learning are : the production of the best player of backgammon, chess and more recently go (algorithm *alphago*), *cartpole* (inverted pendulum on a controlled cart), etc.

To begin with the theory, we need first to understand what are a Markov chain, a Markov Decision Process (MDP) and dynamic programming [3].

A Markov chain is a dynamic system with discrete time $(x_t)_{t \in \mathbb{N}} \in X$, where X is space of states such as :

$$\mathbb{P}(x_{t+1} = x | x_t, x_{t-1}, ..., x_0) = \mathbb{P}(x_{t+1} = x | x_t).$$

Indeed the Markov property states that all the useful information for the future prediction is in the present state. A Markov chain in X is defined by a initial state x_0 and the probability of transition $p(y|x) = \mathbb{P}(x_{t+1} = y | x_t = x)$.

Next come the MDP, which is defined by (X, A, p, r), where :

- X space of states
- A space of actions
- $p(y|x, a)$: probability of transition from a state $x \in X$ to $y \in X$ when the action $a \in A$ is chosen:

$$p(y|x, a) = \mathbb{P}(x_{t+1} = y | x_t = x, a_t = a),$$

- $r(x, a, y)$: reward when passing from x to y using the action a.

We will note here V the value function and π the policy.

Dynamic programming possess two types of iterations : iterations on the values and iterations on the policies. In each of these algorithms a policy sequence is built, beginning with a given initial policy π_0, and every step k, we evaluate the policy π_k and calculate V^{π_k}, then we improve the policy by calculating π_{k+1} given by V^{π_k} :

$$\pi_{k+1}(x) \in \text{argmax}_{a \in A}[r(x, a) + \gamma \sum_y p(y|x, a) V^{\pi_k}(y)]$$

We stop when $V^{\pi_k} = V^{\pi_{k+1}}$. It's important to note that the sequence $(V^{\pi_k})_k$ is a increasing sequence. Therefore, because there is a finite number of possible policies, the termination criteria is obligatory satisfied with k. So we have $V^{\pi_k} = V^*$ where V^* is the optimal value function, therefor π_k is the optimal policy. Some methods of solving are :

- **Direct solving** of the linear system $(I - \gamma P^{\pi})V^{\pi} = r^{\pi}$. That's the Gauss elimination method, but it have a complexity of $O(N^3)$
- **Iteration on the values for a permanent policy** : we iterate on the operator T^{π} (Bellman operator). Considering a given value function V_0, $V_{n+1} = T^{\pi} V_n$. We have then convergence of V_n toward V^{π}. The problem is that the convergence is asymptotical, but the advantage is that it as a lower complexity that the direct solving method

($O(N^2 \frac{log(1/\epsilon)}{log(1/\gamma)})$ for an approximation of ϵ (interesting if γ is not too close from 1)).

- **Monte-Carlo** : we simulate n trajectories $((x_t^i)_{t\geqslant 0})_{1\leqslant i\leqslant n}$, starting from x and following the policy π : x_{t+1}^i $p(.|x_t^i, \pi(x_t^i))$, so :

$$V^\pi(x) \approx \frac{1}{n}\sum_{i=1}^{n}\sum_{t\geqslant 0}\gamma^t r(x_t^i, \pi(x_t^i)).$$

this method is interesting if we want to evaluate a unique state. It has an approximation error of order $O(1/\sqrt{n})$

- **Temporal differences TD(λ)** : This is a smart method for using the trajectories to evaluate all the states crossed by those trajectories, by evaluating the value of a state x, by the sum of the observed temporal differences $r_t + \gamma V(x_{t+1}) - V(x_t)$ at the future instants t, weighted by a "trace" λ. The algorithm of TD(λ) will be treated later.

During the experiments (part Experiments (III)), we used some algorithms (Cross Entropy Method (CEM), REINFORCE and TD(λ)). We will focus here on the TD(λ) method, which we found giving the most reliable results during the experiments. We used a random agent, i.e. selecting actions randomly, in order to test the environment. It is important to note that the Random method was used to test the environment, and the debugging of it.

Temporal Differences TD(λ)

the algorithm TD(λ) [6]: After the observation of a trajectory $(x_0, x_1, ..., x_K = 0)$, we update V_n to the states $(x_k)_0 \leqslant k < K$ following :

$$V_{n+1}(x_k) = V_n(x_k) + \eta_n(x_k)\sum_{l=k}^{K-1}\lambda^{l-k}d_l,$$
$$\text{where } d_l = r^\pi(x_l) + V_n(x_{l+1}) - V_n(x_l).$$

with η_n the learning step, typically $\frac{1}{n}$.

The impact of the temporal differences of future transitions on the estimation of the current state value is controled by λ. TD(λ) is a compromise between :

- **TD**(0) (to estimate the fixed point of the operator of Bellman T^π):

$$V_{n+1}(x_k) = V_n(x_k) + \eta_n(x_k)d_k.$$

- **TD(1)** (to estimate the mean):

$$V_{n+1}(x_k) = V_n(x_k) + \eta_n(x_k)\sum_{l\geqslant k} d_l.$$

the choice of λ :

- $\lambda < 1$ allows to reduce the variance of the estimators compared with $\lambda = 1$.
- $\lambda > 0$ allows to propagate faster the rewards compared with $\lambda = 0$.

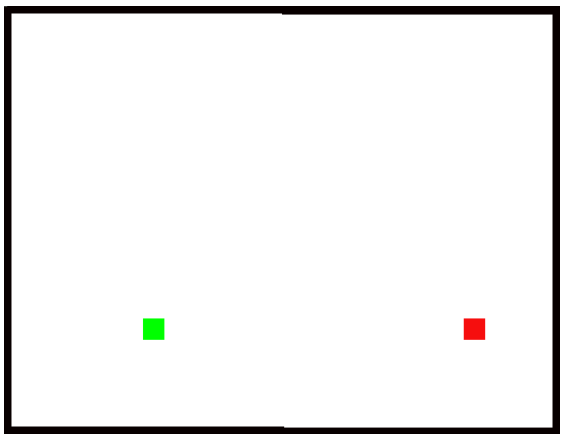

Fig. 2: View of the simulation of the simple determinist environment

the algorithm TD(λ) in the actuated case : We can define the actuated value function when, at each time step, the process is stopped with probability $1-\gamma$ with $0 < \gamma < 1$:

$$V^\pi(x) = \mathbb{E}[\sum_{t\geqslant 0}\gamma^t r(x_t, \pi(x_t))].$$

Algorithm TD(λ) becomes

$$V_{n+1}(x_k) = V_n(x_k) + \eta_n(x_k)\sum_{t\geqslant k}(\gamma\lambda)^{l-k}d_l.$$

Here was a little introduction to the theory of reinforcement learning and a little zoom on the TD(λ) algorithm that we used during the experimentations of our environment.

III. Experiment

We now arrive at the core of this project, which was to develop a learning environment describing the firefighting task in order to improve the policy of the robot and improve its autonomy. For this purpose, we used the toolkit OpenAI Gym, which is a toolkit for developing and comparing reinforcement learning algoriths. We also used the open resource which you can find in the github of torch-twrl (you can follow the link : https://github.com/twitter/torch-twrl). With this resource, you can run already existing environments such as a cartpole environment, an acrobot, board games, etc.

We used those resources to implement and develop our own environment which was the firefighting task.

The mission that we needed to simulate is a mission of search and fight, here representing a firefighter robot. The mission of the robot is to find trees that are on fire, and extinguish them by spreading water on it.

Development of the environment

The first task developed was the most simple one : no moving robot, the only task was to detect if a target was on fire, and then proceed to extinguish it.

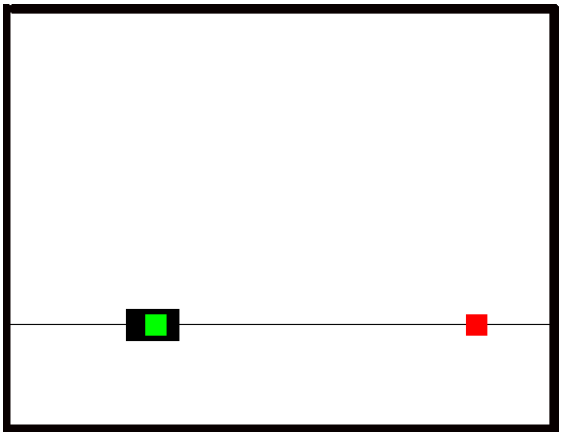

Fig. 3: View of the simulation of the 1D environment

This environment was first developed as determinist. It was an environment where we knew when and where the target will fire up, and see if the robot was able to learn and take measures against. It's important to note that we used the agent/algorithm cited earlier : TD(λ), REINFORCE, Random agent and CEM.
The next step was to turn the determinist environment into a random one, to see if the agents were able to adapt.
After studying the adaptability of the agents to the simplest environment possible for our task, the goal was to develop an environment where there was not the only action of extinguishing the target, but also moving to it.

The first step was still a determinist environment, where we knew when and where the targets will light up.
The second step was to turn the derteminist environment to a random one. The final step was to develop the 2D environment. The first 2D-environment developed was a simple one, with only targets and the robots. It was used to see if the added actions where working accordingly to the real robot.

Then, we developed a more complex environment to make it as close as possible as reality.

To ensure that the environment was close to reality, we discussed the different drawbacks that we needed to implement:

- **the battery** : the energy available for the robot is not infinite, for this, a decrementing counter was implemented, which was decrementing at every action carried out by the robot. There was also a battery zone where the robot could refill up its energy. The battery zone is modeled by the yellow square.

- **the robot's water reserve** : as for the battery, the water that the robot can carry is not unlimited. The water was implemented as a counter which was decremented every time the agent used the action to spill water. This water could be refilled in the water zone which was represented as a blue square.

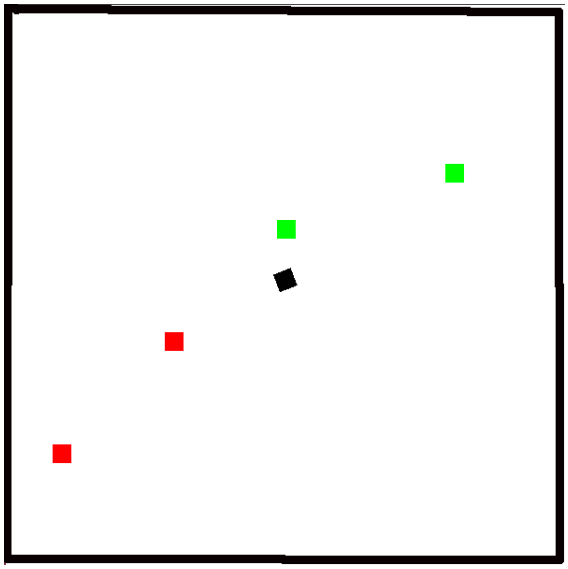

Fig. 4: View of the simulation of the 2D environment

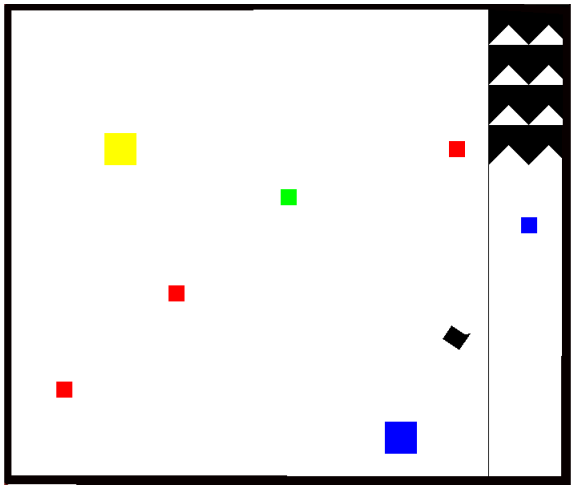

Fig. 5: View of the simulation of the complexified 2D environment

- **the refilling water zone** : represented as a blue square, the water refilling zone could see its water level decreasing (which was viewed by the fading blue color) due to water leaks (which are part of the operator actions.). Those leaks are represented by the square in the right side of the window. Each square represent a level of leaks, and the higher the level is, the more water leaks.

- (the human operator) : the human operator needed to be modeled, as for first, he was designed to not take actions directly on the robot, it was designed to act against the water leaks and fill up the water zone. The operator was designed as a random agent with a fixed random policy.

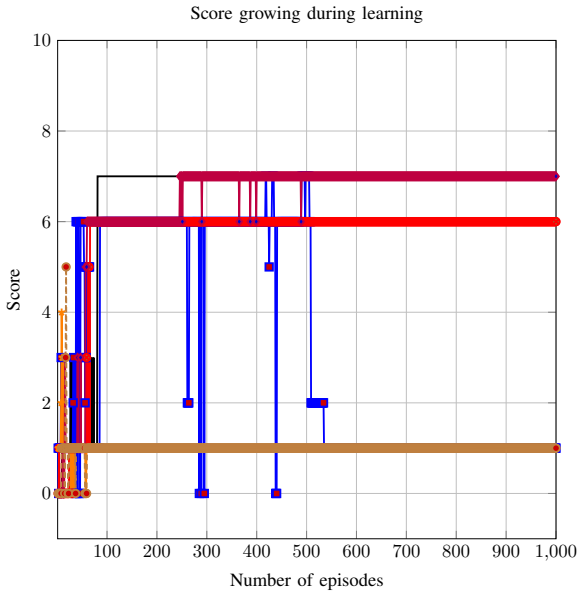

Fig. 6: Score of the deterministic 1D environment for λ

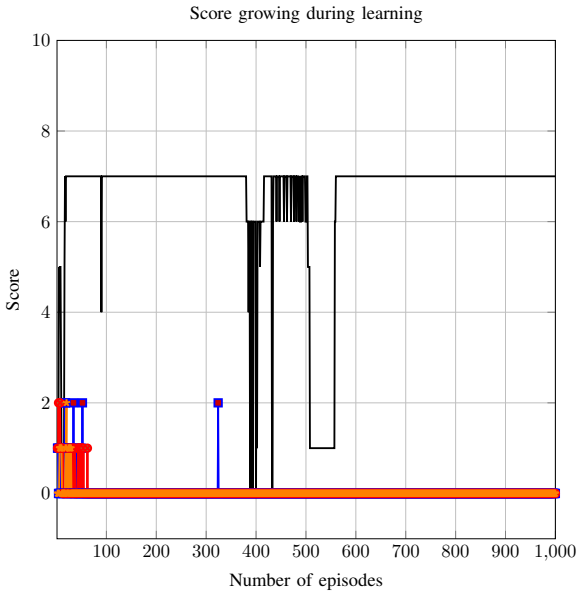

Fig. 7: Score of the deterministic 1D environment for γ

- **the temperature** : The temperature needed also to be modeled, as the robot could not stay to long close to a fire. It was modeled as a counter that was increasing when the robot was in a range from the fire, and decreasing when it was out of this range.

- **the fire** : The fires modeled in the environment where fires that could be extinguished with only one application of water. We discussed the fact that we needed more than one action to take care of it, but it was not implemented ultimately. But the idea still lies.

You can find the environments and videos of the simulations following this address : https://github.com/TVagneron/Reinforcement-learning

Results

In this part, we will display the results of the experiment on the parameters λ, γ and ϵ which affect the learning update or/and the policy. The following results have been runned for a number of 1000 episodes, each episode describing 500 steps. The deterministic environment was tuned to fire one target at a time, every target was lighted up 50 step after the previous was extinguished.

value of λ (fig.6):

- $\lambda = 0.02$, black.
- $\lambda = 0.01$, red.
- $\lambda = 0.005$, purple.
- $\lambda = 0.1$, blue.
- $\lambda = 0.5$, orange.
- $\lambda = 0.8$, brown.

As we can see in the figure, the value of λ affects the reward outcome during the learning process. The more the value go further from the optimum, the less will be the reward. This was to be expected as λ affects the actuating of the process. We can discuss to what extent λ affect the learning process, but it is not the subject of this report, it is to discuss the fine tuning of the parameters in order to optimize the learning process, and optimize the policy. The tuning of λ can be performed mathematically or experimentally. Here we choosed to tune it experimentally. That cas be carry out by a machine, which will hasten the process. We can also take note in the difference of reward for a determinisc environment, which is one of the simplest developed here. It brings up the question of fine tuning for the more complex ones, which have more available actions and space.

Value of γ (fig.7):

- $\gamma = 1$, black.
- $\gamma = 0.7$, red.
- $\gamma = 0.5$, blue.
- $\gamma = 0.3$, orange.

The addition of γ in the learning process also affects the policy (as it appears in both codes). We can see, if γ is different from one, the reward outcome is zero. It was not to be unexpected as γ is a parameter that affect the convergence of the sequence.

Value of ϵ (fig.8):

- $\epsilon = 0.2$, black.
- $\epsilon = 0.1$, red.
- $\epsilon = 0.4$, blue.

ϵ affects the policy, if it is too small, the policy cannot be acualised, resulting in a non-learning behavior, whereas if

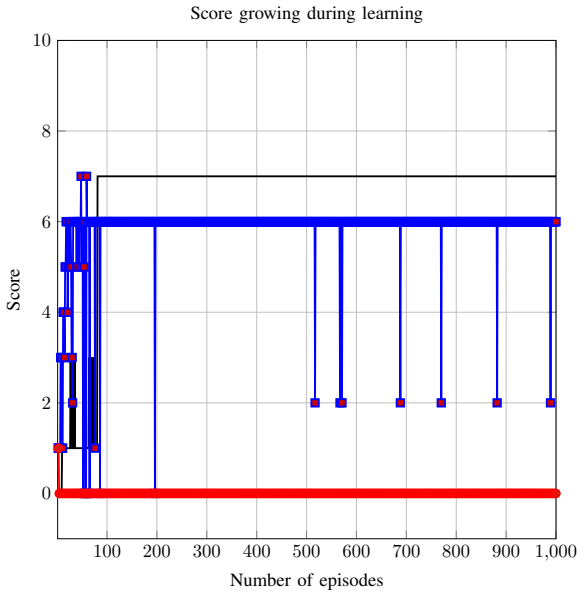

Fig. 8: Score of the deterministic 1D environment for ϵ

the value is too high, the policy will be actualised, but to a different policy than the optimum.

As stated previously, those results are only for the deterministic and one dimensionnal environment. Due to a lack of time, the more complex ones were tested via a random method to verify there functioning. The only tests where we used RL agents were not enough relevant to present them here, but it is important to speak about the optimization of the parameters, and the fact that the development requires to go step by step, as we could let errors slip as we jump over what we could consider as unimportant steps.

We also conducted experimentation on the epsilon decay rate, that affects the processing speed, but the test tests were not relevant as they brought up the issue that will be stated in the issue part later in the report.

Issues

Some problem occured, and this section is here to describe some of them and maybe help to resolve them.

One of the first issue that we encountered when developing our environment was the definition of the observation space, where we needed to define the observables that can be seen by the agent, as for this we defined the states of the targets by a strict binary duo, 0 when the target was extinguished, 1 if it was lighted. To follow strictly our mathematical model, we used a Tuple to define the target states, but Tuple where not accepted for observation space, so we needed to define them as array and boxes. Tuple in openaiGym are a cartesian product between discrete space and continuous space.

The second issue was that when we run our environment, there was an error occurring, stating that the state value was nil. It appears that it was related to a process unable to provide the state at the proper time, resulting in a nil value. This issue was solved by slowing down the process (adding prints to track the error origin solved the problem unexpectedly, but it was also slowed down by adding a wait function in the qfunction). This issue has been observed on two PCs using Ubuntu 14.04. (You can find the issue report following : https://github.com/twitter/torch-twrl/issues/37).

IV. Conclusion

This report is an introduction to the mixed-initiative missions, which are prone to develop in the future. We saw in this report the possibility of using Reinforcement Learning in order to develop a policy for the machine part of the mission, for it to have the optimized behavior in order to complete the human operator. As developed previously, it is important to fine tune the different parameters, as they affect the learning process and the learning speed. On the other hand, the environment developed here are ones that we developed thinking only about the mission. It is always possible to improve them, and there is many ways to do so. Regarding the mixed-initiative missions, we only report here concerning the machine part, which also can be improved, and there is a lot more to do concerning the human part, concerning the eye-tracker, the electroencephalogram, how to handle the data and to adapt them as for the machine part to process them, and also the interaction between the machine and the human. And more generally, the domain of machine learning is a domain where we only began, so there is a lot more to discover.

Acknowledgment

The authors would like to thank Caroline Chanel, because she accepted my participation in this wonderful project.

A huge thank to Nicolas Drougard, without whom i would have been totaly lost, for his patience, his support, and all the mishaps we went through.

References

[1] P. M. Fitts. Human engineering for an effective air-navigation and traffic-control system.
[2] L.Dargent. Profilage pilote par apprentissage automatique pour reconfiguration ihm.
[3] R. Munos. Introduction à l'apprentissage par renforcement.
[4] F. Dehais P. E. U. Souza, C. P. Carvalho Chanel. Momdp-based target search mission taking into account the human operator's cognitive state.
[5] D. Hsu Wee Sun Lee S. C. W. Ong, Shao Wei Png. The human should be part of the control loop?
[6] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
[7] Mai-Hui Le F. Dehais T. Gateau, C. P. Carvalho Chanel. Considering human's non-deterministic behavior and his availability state when designing a collaborative human-robots system.
[8] S. A. Burden M. K. McCourt J. Williard Curtis W. D. Nothwang, R. M. Robinson. The human should be part of the control loop?
[9] M. M. Wilson. Development of online cardiac feedback for control applications.