Project Report: Intrinsically Motivated Goal Exploration Processes in Malmo

Rémy Portelas REMY.PORTELAS@GMAIL.COM

Abstract

This document reports the results of a one-month-long project on using Microsoft Malmo as a platform to study Intrinsically Motivated Goal Exploration Processes (IMGEP) applied to reinforcement learning tasks. More precisely, our objective was to design a multi-goal reinforcement learning environment and then show that using IMGEP as a black-box optimization algorithm for a neural network policy approximator allows to efficiently explore the outcome space.

1. Introduction

This last 3 years' advances in reinforcement learning were mainly achieved through the use of deep neural networks as powerful function approximators. Impressive results were obtained in both discrete action spaces tasks, e.g. playing Atari games from pixels with DQN (Mnih et al., 2015), and continuous action spaces, like learning various control tasks from pixels or joints with DDPG (Lillicrap et al., 2015).

In order to efficiently apply these methods to more complex reinforcement learning tasks, such as robotics, further work is needed in the domain of Exploration. Indeed, current state-of-the-art deep reinforcement learning algorithms usually address exploration through action or parameter perturbation, using simple Gaussian noise or more advanced Ornstein-Uhlenbeck correlated noise.

Recent work (Colas et al., 2018) showed promising results in addressing exploration by sequentially combining methods from the developmental field called Intrinsically Motivated Goal Exploration Processes (Forestier & Oudeyer, 2016; Forestier et al., 2017) with standard deep RL methods.

Parallel to these advances, The Microsoft Malmo platform (Johnson et al., 2016) was released in 2016. Undertaking the growing need for complex benchmarking environments, Malmo allows researchers to efficiently design experiments in Minecraft, an open-ended 3D video-game. Among other things, Malmo allows to design either or both multi-agent and multi-goal environments, which could potentially be procedurally generated.

The aim of this short project was to start exploring how to leverage the Malmo platform to design relevant experiments to analyze IMGEP's performances in complex settings.

The remaining part of this report will be organized as follows: First we will present the variants of the IMGEP framework that were used and how they were combined with a neural network as a policy approximator. We will then describe our test environment named *Malmo Mountain Cart*. Finally we will present and discuss our results.

2. Methods

2.1. Intrinsically Motivated Goal Exploration Processes

IMGEP is a framework originated from the developmental robotics community, formally defined by Forestier and Oudeyer in (Forestier et al., 2017).

To begin with, let us consider an episodic reinforcement learning task. Let Θ be our parameterized policy space and O the outcome space[1] characterizing the result of a policy rollout in a given environment. In practice the outcome of a policy rollout is a vector of features chosen by the experimenter.

IMGEP's steps could be informally described as follows: It begins with a bootstrap phase in which N random policies are generated and executed. The resulting N outcomes are then used to populate a policy-outcome database (θ, o). The second phase consists in sampling a random goal in the outcome space. Finally

Our code along with a demonstration video is available at `https://github.com/rPortelas/malmo_mountain_cart`

[1]Also known as the behavioral descriptor space in the evolutionary community

the third phase is to infer the best policy to reach that goal from our database and, after adding noise to allow the discovery of new outcomes, to execute it and store the resulting policy-outcome pair.

Repeating the second and third phases effectively generates a curriculum of tasks that will efficiently explore the given outcome space, without using any task related reward.

In our experiments, after sampling a new goal, the best corresponding policy is simply inferred by finding the closest outcome in our database based on Euclidean distance. A simple Gaussian noise is then applied to the policy as it proved to be sufficient in Colas' work on combining IMGEP and DDPG(Colas et al., 2018), and which experimental setup is close to ours.

Although the simple method we described above will be part of our study, the main focus of this project was on the modular versions of IMGEP. In this setting, the outcome space is divided by the experimenter in n subspaces and learning happens in n different modules focusing on their respective subspace. This hierarchical approach first chooses a model and then a goal in its corresponding outcome subspace.

For environments composed of multiple subspaces of interests, modular approaches were empirically found to yield better exploration performances than regular (a.k.a flat) IMGEP (Forestier & Oudeyer, 2016). A reason for this success could be that sampling goals in carefully chosen outcome subspaces is often more meaningful than sampling on the entire outcome space, which may often result in sampling unfeasible goals.

For example, consider an object-grasping task using a robotic arm equipped with a gripper. The corresponding outcome space could be discretized trajectories of the arm and the object throughout the episode. In this context, any goal involving to move the object and perform a specific arm trajectory which positions are all far from the object is unfeasible. Splitting this space and sampling goals in the *arm trajectory* or the *object trajectory* subspace would allow to consistently sample meaningful goals.

As in (Forestier et al., 2017), two modular approaches were studied, their difference consisting in how modules were selected at each iteration. Random Babbling simply selects modules randomly whereas Active Model Babbling chooses modules proportionally to their empirical interest. To compute the interest of a module m, a goal g_m is sampled from the outcome subspace O_m, a policy θ is executed, and the sub-outcome o_m is collected. The interest is defined as the absolute value of the mean progress. Progress is computed as $D(g_m, o_m) - D(g_m, o'_m)$, in which o'_m is a previous sub-outcome obtained by executing a policy θ' for a sampled goal g'_m. the function $D(.,.)$ is a normalized euclidean distance.

Using interests to choose modules allows to actively focus learning in relevant outcome subspaces in which improvements are measured and avoids loosing time exploring unfeasible or fully explored subspaces.

2.2. Parameterized policy approximator

For all of our approaches, a neural network with a single hidden layer was used as a parameterized policy approximator. ReLU and Tanh activation functions were used in the hidden and output layers, respectivelly. Inputs were low-dimensional descriptions of the environment state. The network's parameters were chosen during the exploration step and then used for a full rollout in the environment.

An important point of our approach is that the network's parameters are not fine-tuned using gradient-related errors. The idea is more to gather a dataset of neural networks' parameters that were sampled and ran during training.

2.3. Summary of models

Overall, 3 variants of the IMGEP framework and a random control were implemented during this project:

RANDOM: Random policy sampled for each new episode.

F-RGB : Flat Random Goal Babbling, the non-modular version of IMGEP, sampling goals directly in the full outcome space.

M-RMB : Modular Random Model Babbling, a modular version of IMGEP which chooses randomly a module among the n modules defined by the experimenter and then randomly sample a goal in its associated outcome subspace.

M-AMB : Modular Active Model Babbling, in which modules are chosen 20% of the time randomly and 80% proportionally to the empirical mean interest of each module using a soft maximization. Each module only explores (i.e add noise) 4 times out of 5, the last one being used to compute the current interest and update the module's mean interest.

2.4. The Malmo Mountain Cart Environment

An important objective of this project was to creatively leverage Malmo to design an environment well suited for modular IMGEP. To this intent we built what we named the *Malmo Mountain Cart* environment (figure 1), which extends the famous Mountain Car control benchmark in a 3D environment with a multi-goal setting.

In this episodic task with fixed time the agent starts at the bottom left corner of a two-floor arena and is able to act on the environment using 2 continuous action commands: *move* and *strafe*, both using values in $[-1;1]$. *move(1)* moves the agent forward at full speed, *move(-0.1)* moves the agent slowly backward etc. Similarly *strafe(1)* moves the agent left at full speed and *strafe(-0.1)* moves it slowly to the right.

The first challenge of this environment is to learn how to navigate within the arena's boundaries when starting in the confined bottom left corner. If the agent manages to get out of its starting area it may be able to collect items dispatched within the arena by walking in a small radius from them. In our specific instantiation of this environment 5 pieces of bread were placed, 2 of them on the second floor.

A difficult step is to manage to use the stairs in the middle of the arena, which is the only way to reach the second floor. If the agent manages to climb the stairs without falling back to the first floor it may get close enough to the cart to move it along its railroad. If given enough speed, the cart is able to climb the left or right slope and reach one of the golden blocks. The height and width of these slopes were made in such a way that an agent simply moving in front of the cart at full speed will not provide enough speed for the cart to climb the slope. Agents must at least partially support the cart along the track to propel it fast enough to fully climb the slope.

This environment is interesting to study IMGEP's modular approaches since it is composed of a set of linked tasks of increasing complexity. Exploring how to navigate will help to discover the dispatched items and, eventually, will allow to successfully climb the stairs and discover the cart.

2.5. Experimental parameters

As in (Colas et al., 2018), a temperature was empirically tuned for the Tanh output activations of our neural network to ensure that uniformly sampled weights will produce uniformly sampled outputs. Without this feature, since random weigths were sampled between $[-1;1]$ during the bootstrap phase, the networks' outputs tended to saturate, resulting in a pseudo-discretization of the action space in $\{-1,1\}$.

In all of our methods 1000 bootstrap iterations were

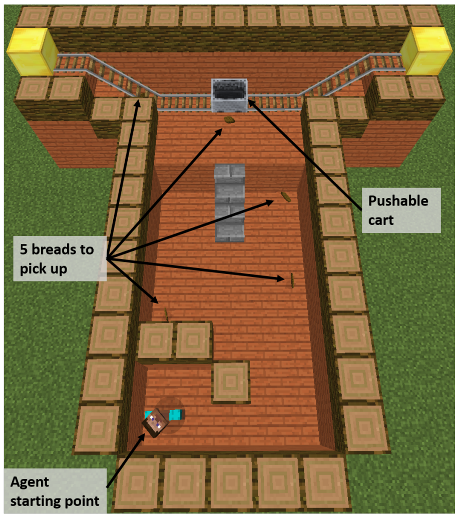


Figure 1. The *Malmo Mountain Cart* environment

performed. After this random phase a Gaussian noise $G(\mu = 0, \sigma = 0.1)$ was added to retrieved policies. For our network architecture this amount of perturbation was empirically found to be high enough to create diversity and low enough to ensure that noised policy were similar to original ones.

During a rollout, which lasts for 7 seconds, actions were choosen at 10Hz. The policy takes a normalized 13D vector composed of 3D positions and velocities of the agent, 1D position and velocity of the cart and a 5D one-hot encoding of the bread (1 if the bread is recovered, 0 otherwise). 2 continuous values between $[-1;1]$ are outputted and used as actions. To avoid the need of any context for our IMGEP methods, the agent, cart and pieces of bread were always reset at the same position when beginning an episode.

The *F-RGB* architecture directly takes the full outcome space which is composed of the final position of the agent (3D) and cart (1D) along with the total number of recovered bread (1D). Although the subspace of final number of bread recovered is discrete (in $[0;5]$), goals were continuous, which does not prevent IMGEP's methods to properly work.

We also considered a variation of this first setup in which distractors were added to the outcome space, as in (Forestier & Oudeyer, 2016). For this case the outcome space is extended with the final position of 2 randomly moving distractors and 2 fixed distractors (all 4 are 3 dimensional).

For IMGEP's modular versions the 5D space was split in 3 subspaces naturally corresponding to the agent position, cart position and total count of recovered bread. In MMC with distractors, each distractor is considered as an additional module.

3. Results

5 runs of 10000 episodes with seeds were performed per exploration architecture on MMC with and without distractors. As *RANDOM* is not affected by distractors since it does not use the outcome space, we only performed 5 runs used as control in both environment configurations. Each run corresponds to 19.5 hours of gameplay. Using a regular laptop with GPU we were able to launch up to 5 runs in parallel and to reduce the execution of a run to 8 hours of wall-clock time.

In this section we will compare our different approaches on exploration and competence measures on MMC's 3 outcome subspaces: the agent's final position (the agent space for short), the cart's final position (cart space) and the number of recovered pieces of bread (bread space).

3.1. Exploration

Running *RANDOM* for a total of 50000 episodes allowed us to analyze the difficulty of the environment in terms of exploration. Agents following random policies were found to have less than 2% chances of finding 1 or 2 pieces of bread. Finding 3 of them or touching the cart have both less than 0.01% chances of being observed. 4 pieces of bread were found once across the 50000 episodes and never have all 5 of them been recovered nor the cart pushed up to a gold block.

Figure 2 shows a qualitative example of exploration coverage after 10000 training episodes for each architecture, without distractors. One can clearly see that *RANDOM* agent's final positions are not very diverse, most of them end up stuck on a wall close to the starting point. In this run it managed to touch the cart once, pushing it slightly to the left. Comparatively, *F-RGB* performs much better and managed to push the cart up both slopes. Compared to *F-RGB* modular methods better explored the second floor and the cart space than the flat architecture.

3.1.1. discovered behaviors

In all of the 10 runs performed (5 runs per environment configuration), IMGEP architectures always discovered how to push the cart up to the left or right gold block and how to retrieve 0 to 4 pieces of bread. Without distractors *AMB* and *F-RGB* discovered how to push the cart up in both sides in 3/5 runs and up to 4/5 for *RMB*. With distractors only *RMB* managed to push it up the left and right side in 1/5 runs.

Although distractors negatively affected the exploration of the cart space, it was easier for IMGEP architectures to find 5 pieces of bread in MMC with distractors. Without distractors *F-RGB* is the only architecture who discovered how to retrieve 5 pieces of bread in the map, which is observed only in 1/5 runs. With distractors, both *F-RGB* and *RMB* found it in 1/5 runs, and *AMB* discovered it in 2/5 runs.

3.1.2. Grid-Cell exploration

Grid-cell exploration was chosen as the measure to statistically evaluate our IMGEP architectures. For the agent space we discretize in 300 cells a rectangle including the arena. 80 cells are used for the cart. Regarding the number of recovered bread, since there is only 6 possible outcomes (0 to 5 pieces of bread), we decided to use the number of possible combinations of recovered bread (32) to measure exploration, which is more informative.

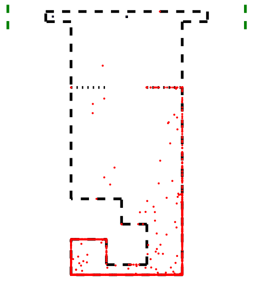
Figure 3 shows the evolution of grid-cell exploration for the agent, cart and bread spaces, with and without distractors. *RANDOM* manages to slowly explore the agent and bread spaces but the cart space exploration remains close to 0 as it is very rare to touch the cart when sampling random policies. IMGEP architectures match *RANDOM* performances during their bootstrapping phase and then quickly increase and outperform it in every outcome subspace as they start leveraging their database of policy-outcome pairs.

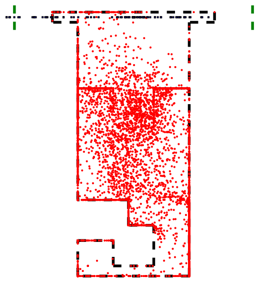
In MMC without distractors, the exploration of the agent space is similar for *F-RGB*, *RMB* and *AMB*, although the latter is lower by a small margin throughout training (reaching 26.6% against 27.6% for *F-RGB* and *RMB*). In the more challenging cart space, modular methods perform significantly better than flat architectures, with *RMB* and *AMB* respectively reaching 81% and 79% against 68% of exploration coverage for *F-RGB*. With distractors the exploration of *RMB* in the agent space is slowed as it often chooses to sample goals in the distractors outcome subspaces. Comparatively, *AMB* learned to ignore these unpromising outcome subspaces and therefore maintains its performances. Interestingly, *F-RGB*'s performances remains unchanged.

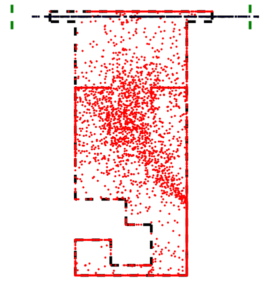
3.2. Interests in Active Model Babbling

The interests evolutions of all of the *AMB* runs are left for consultation in the appendix (figure 5 and 6).

Typical runs for MMC with and without distractors are displayed in figure 4. For both environments one


(a) *RANDOM*


(b) *F-RGB*


(c) *RMB*

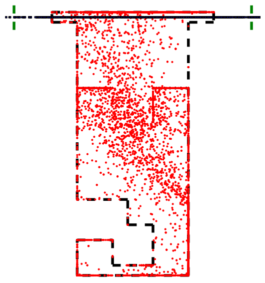
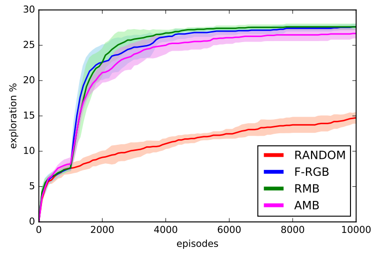
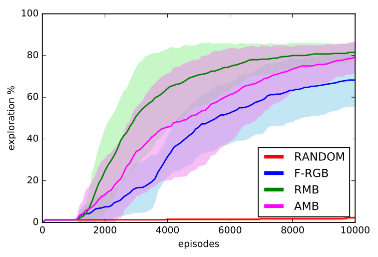

(d) *AMB*

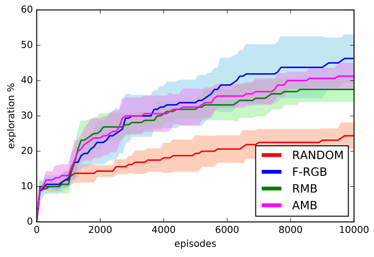
Figure 2. End positions of the agent (red dots) and the cart (blue squares) after 10000 episodes. The area delimited by black dashes is the agent's reachable space within the arena. The left and right green dashes shows the point at which the cart fully climbs the slope and stops near the gold block.



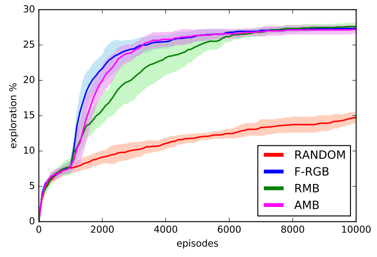
(a) Agent final position



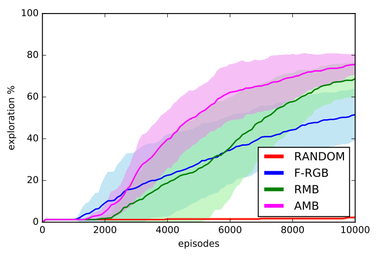
(b) Cart final position



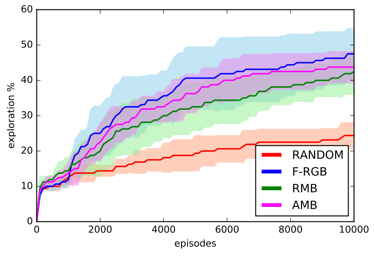
(c) Bread combinations



(d) Agent (with distractors)

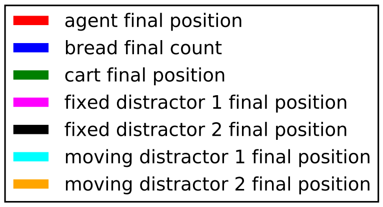


(e) Cart (with distractors)

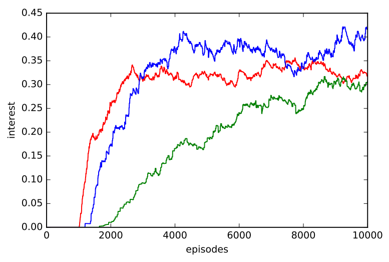


(f) Bread (with distractors)

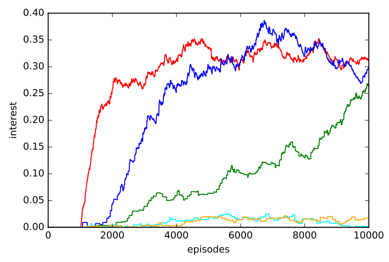
Figure 3. Averaged grid-cell exploration evolution of Flat Random Goal Babbling, Random Model Babbling and Active Model Babbling (along with a control sampling random policy's parameters) throughout training. (a), (b) and (c): MMC environment. (c) (d) and (e): MMC environment with added distractors.



(a)



(b) MMC



(c) MMC with distractors

Figure 4. Evolution of modules' interest in *AMB* when run on MMC with (b) and without (c) distractors. Following empirical interests for module selection effectively creates an intrinsic learning curriculum by shifting exploration focus on tasks where progress is maximal.

can see that the agent and bread spaces are first detected to be where the highest progress is made. In most runs (9/10) they remain the top 2 outcome subspaces in terms of interest. In these two specific runs their interest starts to stabilize after around 5000 episodes, which is understandable since the exploration in the agent and bread spaces slows down at this point (see figure 3).

Once the cart is first discovered and moved (after around 2000 episodes in those two cases), the interest of its corresponding module slowly increases throughout the run, a phenomenon starting at around 2000 to 4000 episodes for all runs.

The overall shapes of the interests of the agent, cart and bread modules were not disturbed by adding distractors. Fixed distractors have an interest strictly equal to 0 as no progress is ever made in their outcome subspace. *AMB*'s interest computation is also robust to random distractors whose progress is randomly positive or negative, as one can see that their interests remains close to 0 throughout training.

3.3. Exploitation

Competence was measured in each outcome subspace as the averaged distance error on a test set of random goals composed of 850 final agent positions, 100 final cart positions and 50 bread final counts. This same test set was used on all of the considered architectures after 10000 training episodes. During test phase, for each goal, the outcome-pair database was simply used by a nearest neighbor algorithm to retrieve the previous policy which associated outcome was the closest from the given test goal.

The obtained competence errors are summarized in table 1. As expected, IMGEP architectures perform better than the naive *RANDOM* approach in every outcome subspaces. Modular and flat IMGEP architectures perform similarly well in the agent space, regardless of the use of distractors, their mean errors ranging from 0.07 for *F-RGB* in MMC without distractors to 0.05 for *RMB* with distractors and *AMB* without distractors. Apart from these obvious observations, further analysis becomes risky as the non-determinism of the Malmo platform made our competence evaluation method unreliable.

We discovered that re-running our test set on the same policy-outcome pair database often resulted in significantly different results, mainly for the cart and bread spaces, in which the slightest environmental difference during an episode might prevent the agent to get close enough to a bread or push the cart properly. For example, when considering the goal of moving the cart left, an agent unable to reach the cart might get a lower error than an agent who discovered how to do it during training but pushed it on the right side during testing due to non-determinism and therefore get a high error when it should not.

To unformaly assess the gravity of the issue, we tried to re-run a policy that was discovered to allow an agent to recover all 5 breads during training. In this case, more than 15 attempts were necessary for the agent to correctly reproduce the behavior it performed during exploration.

This issue was not solved in time but our guess is that increasing the test set size, especially for bread and cart goals, might alleviate the variance introduced by the non-determinism of the Malmo platform.

Table 1. Averaged competence errors on a test set of 1000 random goals (850 final agent positions, 100 final cart positions and 50 bread final counts). Format is *mean* ± *standard deviation.*

Method	Distractors?	Agent	Cart	Bread
RANDOM	No Yes	0.14± 0.05	0.43± 0.05	0.27± 0.05
F-RGB	No	0.07± 0.01	0.21± 0.06	0.21± 0.02
	Yes	0.06± 0.01	0.19± 0.04	0.18± 0.05
RMB	No	0.06± 0.04	0.22± 0.06	0.23± 0.08
	Yes	0.05± 0.01	0.31± 0.09	0.24± 0.04
AMB	No	0.05± 0.02	0.20± 0.06	0.19± 0.05
	Yes	0.06± 0.01	0.23± 0.01	0.21± 0.03

4. Conclusion and Discussion

Our results in terms of exploration put forward an interesting question: why do distractors helped to better explore the bread space but hindered the cart's space exploration ?

A possible explanation could be that adding distractors lead IMGEP approaches to genererate more unfeasible goals, leading to select and add noise to a more diverse set of policies. Distractors therefore increased the randomness of the exploration, hindering exploration in the hard-to-reach cart space but increasing discovery chances in the bread space which, from our point of view, might benefit from a broader exploration. The idea behind this last claim is that, for the bread space, selecting a wider range of policies to explore on allows to discover a diverse set of trajectories leading to find new combinations of recovered pieces of bread. In contrast, making new discoveries in the cart space requires to find the small end-of-trajectories variations that will push the cart at different positions.

These hypothesized characteristics of the cart and bread spaces might also explain why *F-RGB* is better at exploration in the bread space than modular architectures but worse in the cart space. Our idea is that in flat IMGEP architectures goals are sampled in the full outcome space which might results in trying to reach unfeasible goals, leading to select a wider range of policies than in *RMB* and *AMB*. Comparatively, modular architectures more effectively sample feasible goals by focusing on outcome subspaces, leading them to only select policies among the restrained set of promising ones per subspace, which might not generate as much diversity than flat architectures in the bread space but will allow to efficiently find variations of successful policies in the cart space.

We showed that Active Model Babbling was able to successfully focus its exploration on relevant tasks by selecting modules at each episode according to their interest, effectively avoiding to spend time trying to move the cart when it was not yet discovered or trying to learn the unlearnable by sampling goals in the distractors' spaces. As expected, *AMB* performed better than *RMB* in the cart and bread space with distractors. However *RMB* obtained better exploration performances than *AMB* for the cart space in MMC without distractors. This indicates that MMC's regular outcome space is too simple to waste exploration time trying to compute interests for active module choice, which was done in 1/5 episodes in our case.

Limits and future work. An obvious limit of this work, due to the one-month time constraint of the project, is the lack of statistical analysis in terms of exploration and especially for competence analysis. It would be interesting to see if the obtained results still hold after an in-depth analysis with more, and potentially longer runs.

Another interesting next step after this project would be to extend the MMC environment to increase complexity by forcing agent to first learn a complex combination of nested tool use before being able to swing the cart. This could be easily achieved in Malmo which allows to easily customize environments. An example of such an extension is shown in the appendix, figure 7. This extended MMC environment might better reveal *AMB*'s ability to efficiently organise exploration.

In this project, distractors where used only in the outcome space. It would be interesting to have those distractors added to the low dimensional input of the neural network used as policy approximator, which would make the MMC environment much harder, since one would need to find policies which outputs are invariant to the noise introduced by distractors. This harder setting would be closer to real-life scenarios where state observations are often noisy.

This time-constrained project focused on evaluating IMGEP approaches for the Malmo Mountain Cart environment, however further work should be dedicated to test other reinforcement learning methods in MMC to compare our results. Among others, DDPG and its variation using IMGEP called GEP-PG would be of particular interest.

Finally, another line of investigation would be to use the Malmo platform to test if recent work (Péré et al., 2018) on autonomous learning of outcome spaces using deep representation learning could scale to a challenging environment such as MMC.

References

Colas, Cédric, Sigaud, Olivier, and Oudeyer, Pierre-Yves. GEP-PG: decoupling exploration and exploitation in deep reinforcement learning algorithms. *CoRR*, abs/1802.05054, 2018.

Forestier, Sébastien and Oudeyer, Pierre-Yves. Modular active curiosity-driven discovery of tool use. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016/10 2016.

Forestier, Sébastien, Mollard, Yoan, and Oudeyer, Pierre-Yves. Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*, abs/1708.02190, 2017.

Johnson, Matthew, Hofmann, Katja, Hutton, Tim, and Bignell, David. The malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 4246–4247, 2016.

Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836.

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In *International Conference on Learning Representations*, 2018.

A. Interest curves and potential extension to MMC

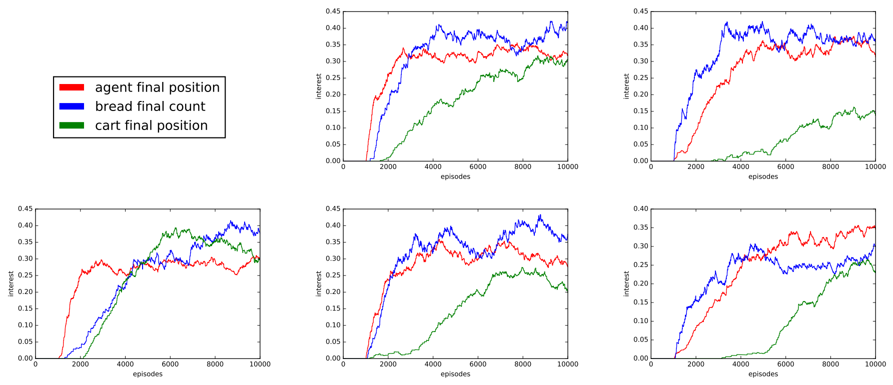


Figure 5. Evolution of modules' interest in 5 runs of *AMB* in MMC.

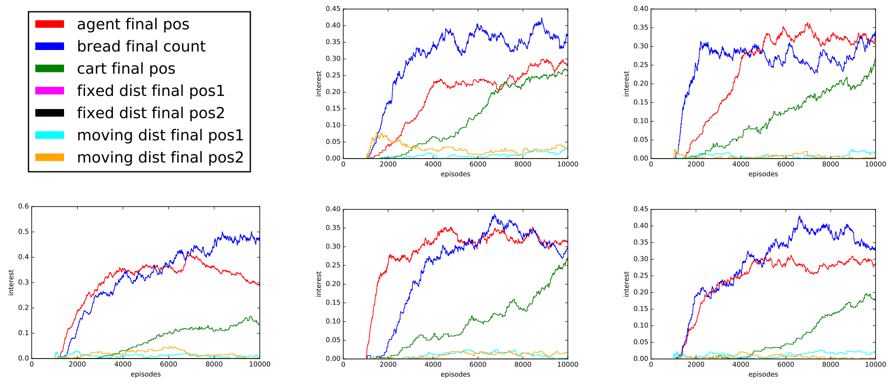


Figure 6. Evolution of modules' interest in 5 runs of *AMB* in MMC with two moving and two fixed distractors.

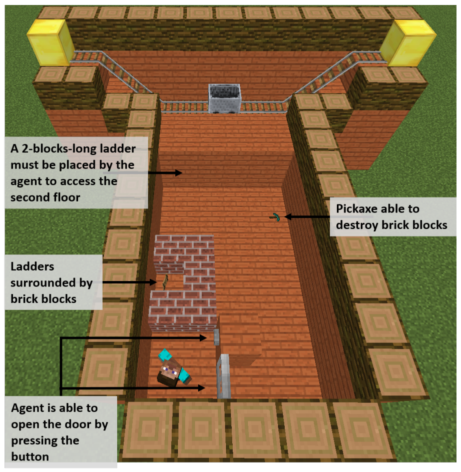

A 2-blocks-long ladder must be placed by the agent to access the second floor
Pickaxe able to destroy brick blocks
Ladders surrounded by brick blocks
Agent is able to open the door by pressing the button

Figure 7. Extended Malmo Mountain Cart (EMMC). A possible extension of the MMC environment in which accessing to the second floor necessitates a complex nested tool use process: Agent must first get out of its starting room by pressing a button to open a door. A second step is then to collect a pickaxe allowing the agent to destroy the brick blocks and collect ladders. If correctly placed on the wall and successfully used, agents may access the second floor. This environment might be interesting to use in future work to better exhibit the power of Active Model Babbling compared to Random Model Babbling.