Learning a Set of Interrelated Tasks by Using Sequences of Motor Policies for a Strategic Intrinsically Motivated Learner

Nicolas Duminy, Sao Mai Nguyen, Dominique Duhaut

N. Duminy, S. M. Nguyen and D. Duhaut, "Learning a Set of Interrelated Tasks by Using Sequences of Motor Policies for a Strategic Intrinsically Motivated Learner," 2018 Second IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, US, 2018, pp. 288-291. doi:10.1109/IRC.2018.00061

Learning a set of interrelated tasks by using sequences of motor policies for a strategic intrinsically motivated learner

Nicolas Duminy[1] Sao Mai Nguyen[2] Dominique Duhaut[1]

***Abstract*—We propose an active learning architecture for robots, capable of organizing its learning process to achieve a field of complex tasks by learning sequences of motor policies, called Intrinsically Motivated Procedure Babbling (IM-PB). The learner can generalize over its experience to continuously learn new tasks. It chooses actively what and how to learn based by empirical measures of its own progress. In this paper, we are considering the learning of a set of interrelated tasks outcomes hierarchically organized.**

We introduce a framework called "procedures", which are sequences of policies defined by the combination of previously learned skills . Our algorithmic architecture uses the procedures to autonomously discover how to combine simple skills to achieve complex goals. It actively chooses between 2 strategies of goal-directed exploration: exploration of the policy space or the procedural space. We show on a simulated environment that our new architecture is capable of tackling the learning of complex motor policies, to adapt the complexity of its policies to the task at hand. We also show that our "procedures" framework helps the learner to tackle difficult hierarchical tasks.

I. Introduction

Taking a developmental robotic approach [1], we combine the approaches for active motor skill learning of goal-oriented exploration and strategical learning to learn multiple complex and interrelated tasks. Our algorithm is able to learn a mapping between a continuous space of parametrized tasks (also referred to as outcomes) and a space of parametrized motor policies (sometimes referred to as actions).

A. Active motor skill learning of multiple tasks

Classical techniques based on Reinforcement Learning [2] [3] still need an engineer to manually design a reward function for each task. Intrinsic motivation (IM), which triggers curiosity in humans according to developmental psychology [4], was introduced in highly-redundant robots to make them learn a wider range of tasks, through goal-babbling [5] [6].

However, with higher outcome space dimensionalities, their efficiency drops [7] due to the curse of dimensionality.

B. Strategic learning

Approaches where the learner chooses both what (which outcome to focus on) [5] and how (which strategy to use) [9] to learn are called strategic learning [8]. They aim at enabling an autonomous learner to self-organize its learning process.

The research work presented in this paper is partially supported by the EU FP7 grant ECHORD++ KERAAL and by the the European Regional Fund (FEDER) via the VITAAL Contrat Plan Etat Region

[1] Nicolas Duminy and Dominique Duhaut are with Université Bretagne Sud, Lorient, France. nicolas.duminy@telecom-bretagne.eu and dominique.duhaut@univ-ubs.fr

[2] Sao Mai Nguyen is with IMT Atlantique, Lab-STICC, UBL, F-29238 Brest, France. nguyensmai@gmail.com

The problem was introduced and studied in [8], and implemented for an infinite number of outcomes and policies in continuous spaces by the SGIM-ACTS algorithm [15]. This algorithm organizes its learning process, by choosing actively both which strategy to use and which outcome to focus on. It relies on the empirical evaluation of its learning progress. It could choose among autonomous exploration driven by IM and low-level imitation of one of the available human teachers to learn more efficiently. It showed its potential to learn on a real high dimensional robot a set of hierarchically organized tasks [11], so we inspire from it to learn complex motor policies.

C. Learning complex motor policies

In this article, we tackle the learning of complex motor policies, which we define as sequences of primitive policies.

We wanted to enable the learner to decide autonomously the complexity of the policy necessary to solve a task, so we discarded via-points [3]. Options [12] are temporally abstract actions built to reach one particular task. They have only been tested for discrete tasks and actions, where a small number of options were used, whereas our new proposed learner is to be able to create an unlimited number of complex policies.

As we aim at learning a hierarchical set of interrelated complex tasks, our algorithm could use this task hierarchy (as [13] did to learn tool use with primitive policies only), and try to reuse previously acquired skills to build more complex ones. [14] showed that building complex actions made of lower-level actions according to the task hierarchy can bootstrap exploration by reaching interesting outcomes more rapidly.

We adapted SGIM-ACTS to learn complex motor policies of unlimited size. We developed a new mechanism called "procedures" (see Section II-B) which proposes to combine known policies according to their outcome. Combining these, we developed a new algorithm called Intrinsically Motivated Procedure Babbling (IM-PB) capable of taking task hierarchy into account to learn a set of complex interrelated tasks using adapted complex policies. We will describe an experiment, on which we have tested our algorithm, and we will present and analyze the results.

II. Our approach

Inspired by developmental psychology, we propose a strategic learner driven by IM. This learner discovers the task hierarchy and reuses previously learned skills while adapting the complexity of its policy to the complexity.

In this section, we formalize our learning problem and explain the principles of IM-PB.

A. Problem formalization

In our approach, an agent can perform policies π_θ, parametrized by $\theta \in \Pi$. Those policies induce outcomes in the environment, parametrized by $\omega \in \Omega$. The agent is then to learn the mapping between Π and Ω: it learns to predict the outcome ω of each policy π_θ (the forward model M), but more importantly, it learns which policy to choose for reaching any particular outcome (an inverse model L). The outcomes ω are of various dimensionality and be split in task spaces $\Omega_i \subset \Omega$.

The policies consist of a succession of primitives (encoded by the same set of parameters $\theta \in \Pi$) that are executed sequentially by the agent. Hence, policies also are of different dimensionality and are split in policy spaces $\Pi_i \subset \Pi$ (where i corresponds to the number of primitives).

B. Procedures

As this algorithm tackles the learning of complex hierarchically organized tasks, exploring and exploiting this hierarchy could ease the learning of the more complex tasks. We define procedures as a way to encourage the robot to reuse previously learned skills, and chain them to build more complex ones. More formally, a procedure is built by choosing two previously known outcomes ($t_i, t_j \in \Omega$) and is noted $t_i \boxplus t_j$.

Executing a procedure $t_i \boxplus t_j$ means building the complex policy π_θ corresponding to the succession of both policies π_{θ_i} and π_{θ_j} and execute it (where π_{θ_i} and π_{θ_j} reach best t_i and t_j respectively). As t_i and t_j are generally unknown from the learner, the procedure is updated before execution (see Algo. 1) to subtasks t_1 and t_2 which are feasible by the learner according to its current skill set.

Algorithm 1 Procedure modification before execution

```
Input: $(t_i, t_j) \in \Omega^2$
Input: inverse model $L$
  $t_1$ ←Nearest-Neighbour($t_i$)
  $t_2$ ←Nearest-Neighbour($t_j$)
  $\pi_{\theta_1} \leftarrow L(t_1)$
  $\pi_{\theta_2} \leftarrow L(t_2)$
  return $\pi_\theta = \pi_{\theta_1}\pi_{\theta_2}$
```

C. Intrinsically Motivated Procedure Babbling

The IM-PB algorithm (see Algo. 2) learns by episodes, where an outcome $\omega_g \in \Omega$ to target and an exploration strategy σ have been selected.

In an episode under the policy space exploration strategy, the learner tries to optimize the policy π_θ to produce ω_g by choosing between random exploration of policies and local optimization, following the SAGG-RIAC algorithm [5] (Goal-Directed Policy Optimization(ω_g)). Local optimization uses local linear regression.

In an episode under the procedural space exploration strategy, the learner builds a procedure $t_i \boxplus t_j$ such as to reproduce the goal outcome ω_g the best (Goal-Directed Procedure Optimization(ω_g)). It chooses either random exploration of procedures (which builds procedures by generating two subtasks at random) when the goal outcome is far from any previously reached one, or local procedure optimization, which optimizes a procedure using local linear regression. The procedure built is then modified and executed, using Algo. 1.

After each episode, the learner stores the policies and modified procedures executed along with their reached outcomes in its episodic memory. It computes its competence in reaching the goal outcome ω_g by comparing it with the outcome ω it actually reached (using normalized Euclidean distance $d(\omega, \omega_g)$). Then it updates its interest model according to the progress $p(\omega_g)$, which is the derivate of the competence, it has made (including the outcome spaces reached but not targeted) in order to choose the strategy and task in the next episode. The interest *interest*(ω, σ) of each outcome added depends on both the progress $p(\omega)$ made and the cost $K(\sigma)$ of the strategy used: *interest*$(\omega, \sigma) = p(\omega)/K(\sigma)$. The outcomes reached and the goal are added in their corresponding region, which is then split when exceeding a fixed number of points to discriminate the regions of high and low interest for the learner. The method used is described in [15].

Algorithm 2 IM-PB

```
Input: the different strategies $\sigma_1, ..., \sigma_n$
Initialization: partition of outcome spaces $R \leftarrow \bigsqcup_i \{\Omega_i\}$
Initialization: episodic memory Memo ← $\varnothing$
  loop
    $\omega_g, \sigma$ ← Select Goal Outcome and Strategy($R$)
    if $\sigma$ = Autonomous exploration of procedures strategy then
      Memo ← Goal-Directed Procedure Optimization($\omega_g$)
    else
      $\sigma$ = Autonomous exploration of policies strategy
      Memo ← Goal-Directed Policy Optimization($\omega_g$)
    end if
    Update $L^{-1}$ with collected data Memo
    $R$ ← Update Outcome and Strategy Interest Mapping($R$,Memo,$\omega_g$)
  end loop
```

The choice of strategy and goal outcome is based on the empirical progress measured in each region R_n of the outcome space Ω, as in [11].

When the learner computes nearest neighbours to select policies or procedures to optimize (when choosing local optimization in any autonomous exploration strategies and when refining procedures), it actually uses a performance metric (1) which takes into account the cost of the policy chosen:

$$perf = d(\omega, \omega_g)\gamma^n \qquad (1)$$

where $d(\omega, \omega_g)$ is the normalized Euclidean distance between the target outcome ω_g and the outcome ω reached by the policy, γ is a constant and n is equal to the size of the policy (the number of primitives chained).

III. Experiment

We designed an experiment with a simulated robotic arm, which can move and interact with objects. It can learn an infinite number of tasks, organized as 6 types of tasks. The robot can perform complex policies of unrestricted size.

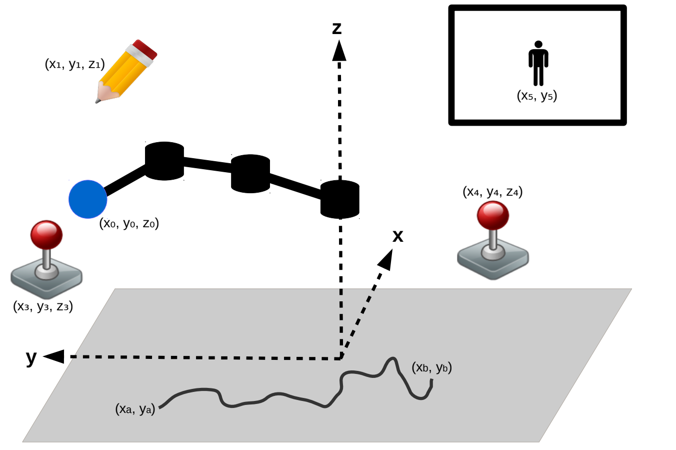

Fig. 1: Experimental setup: a robotic arm, can interact with the different objects in its environment (a pen and two joysticks). Both joysticks enable to control a video-game character (represented in top-right corner). A grey floor limits its motions and can be drawn upon using the pen (a possible drawing is represented).

A. Simulation setup

Fig. 1 shows the experimental setup (delimited by $(x, y, z) \in [-1; 1]^3$). The learning agent is a planar robotic arm of 3 joints (each link measures 0.33), with the based anchored in the center of the horizontal plan. It can rotate around the z-axis and change its vertical position. The robot can grab objects by hovering its arm tip (blue in Fig. 1) close to them, which position is noted (x_0, y_0, z_0). It interacts with:

- Pen: position noted (x_1, y_1, z_1), can be moved and draw on the floor, broken if forcing to much on the floor;
- Floor which limits motions to $z > -0.2$;
- Drawing: last continuous line made when the pen moves on the floor, delimited by first (x_a, y_a) and last (x_b, y_b) point, if the pen is functional;
- Joysticks: two joysticks can be moved inside their own cubic-shape volume and control a video-game character, released otherwise, normalized positions respectively at (x_3, y_3, z_3) and (x_4, y_4, z_4);
- Video-game character: on a 2D-screen set by the joysticks refreshed only at the end of a primitive policy execution for manipulated joystick, position (x_5, y_5) set by joystick 1 x-axis and joystick 2 y-axis respectively.

The robot can one object at once. Touching an other breaks it, releasing both objects. It always starts from the same position before executing a policy, and primitives are executed sequentially without getting back to this initial position. Whole complex policies are recorded with their outcomes, but each step of the complex policy execution is recorded as well.

B. Experiment variables

1) Policy spaces: The motions of each of the three joints of the robot are encoded using a one-dimensional Dynamic Movement Primitive (DMP). We are using the original form of the DMP from [16] and we keep the same notations. Each of these one-dimensional DMP a_i (ordered by joint from base to tip) is encoded using its end position $g^{(i)}$, and three basis functions for the forcing term, parametrized by their weights $(\omega_0^{(i)}, \omega_1^{(i)}, \omega_2^{(i)})$. A primitive motor policy is simply the concatenation of those DMP parameters and the fixed vertical position of the arm during the motion z:

$$\theta = (a_0, a_1, a_2, z) \tag{2}$$

$$a_i = (\omega_0^{(i)}, \omega_1^{(i)}, \omega_2^{(i)}, g^{(i)}) \tag{3}$$

When combining two or more primitive policies $(\pi_{\theta_0}, \pi_{\theta_1}, ...)$, in a complex policies π_θ, the parameters $(\theta_0, \theta_1, ...)$ are simply concatenated together from the first primitive to the last.

2) Task spaces: The task spaces the robot learns are hierarchically organized and defined as: $\Omega_0 = \{(x_0, y_0, z_0)\}$, $\Omega_1 = \{(x_1, y_1, z_1)\}$, $\Omega_2 = \{(x_a, y_a, x_b, y_b)\}$, $\Omega_3 = \{(x_3, y_3, z_3)\}$, $\Omega_4 = \{(x_4, y_4, z_4)\}$ and $\Omega_5 = \{(x_5, y_5)\}$.

C. Evaluation method

To evaluate our algorithm, we created a benchmark linearly distributed across the Ω_i, of 27,600 points. The evaluation consists in computing mean Euclidean distance between each of the benchmark outcomes and their nearest neighbour in the learner dataset. This evaluation is repeated regularly.

Then to asses our algorithm efficiency, we compare its results of algorithms: RandomPolicy (random exploration of Π) , SAGG-RIAC (exploration of Π guided by IM), Random-PB (random exploration of policies and procedures), IM-PB (exploration of the procedural space and the policy space, guided by IM).

Each algorithm was run 5 times for 25,000 iterations (complex policies executions). The meta parameter was: $\gamma = 1.2$.

IV. Results

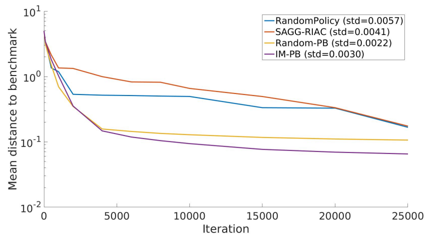

Fig. 2: Evaluation of all algorithms (standard deviation shown in caption)

Fig. 2 shows the global evaluation of all tested algorithms, which is the mean error made by each algorithm to reproduce the benchmarks with respect to the number of complete complex policies tried. Random-PB and IM-PB owing to procedures have lower errors than the others even since the beginning. Indeed, they perform better than their downgrades without procedures, RandomPolicy and SAGG-RIAC.

On each individual outcome space (Fig. 3), IM-PB outperforms the other algorithms. The comparison of the learners without procedures (RandomPolicy and SAGG-RIAC) with the others shows they learn less on any outcome space but Ω_0 (reachable using single primitives, with no subtask) and especially for Ω_1, Ω_2 and Ω_5 which were the most hierarchical

N. Duminy, S. M. Nguyen and D. Duhaut, "Learning a Set of Interrelated Tasks by Using Sequences of Motor Policies for a Strategic Intrinsically Motivated Learner," 2018 Second IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, US, 2018, pp. 288-291. doi:10.1109/IRC.2018.00061

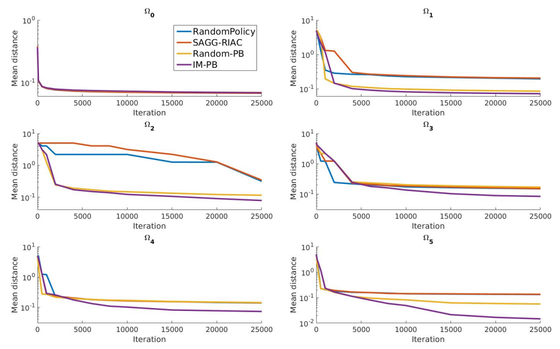

Fig. 3: Evaluation of all algorithms per outcome space (for Ω_0, all evaluations are superposed)

in this setup. So the procedures helped when learning any potentially hierarchical task in this experiment.

We wanted to see if our IM-PB learner adapts the complexity of its policies to the working task. We draw 1,000,000 goal outcomes for each of the Ω_0, Ω_1 and Ω_2 subspaces (chosen because they are increasingly complex) and we let the learner choose the known policy that would reach the closest outcome. Fig. 4 shows the results of this analysis.

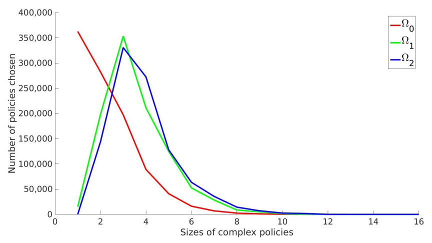

Fig. 4: Number of policies selected per policy size for three increasingly more complex outcome spaces by the IM-PB learner

As we can see on those three interrelated outcome subspaces (Fig. 4), the learner is capable to adapt the complexity of its policies to the outcome at hand. It chooses longer policies for Ω_1 and Ω_2 (size 3 and 4 compared to size 1 for Ω_0). Our learner is capable to correctly limit the complexity of its policies instead of being stuck into always trying longer and longer policies. However, the learner did not increase its policies complexity from Ω_1 to Ω_2, as we hoped.

V. Conclusion and Future work

With this experiment, we show the capability of IM-PB to tackle the learning of a set of multiple interrelated complex tasks. It successfully uses complex motor policies to learn a wider range of tasks. Though it was not limited in the size of policies it could execute, the learner shows it could adapt the complexity of its policies to the task at hand.

The procedures greatly improved the learning capability of autonomous learners, as we can see by the difference between the Random-PB and IM-PB learners and the RandomPolicy and SAGG-RIAC ones. Our IM-PB shows it is capable to use procedures to exploit both the task hierarchy of this experimental setup and previously learned skills.

However this new framework of procedures could be better exploited, if it could be recursive (defined as a binary tree structure), allowing the refinement process to select lower-levels procedures as one of the policy component. This process could also be used inside the strategical decisions made by the learner when selecting what and how to learn. This strategical choice could also be recursive, allowing the learner to optimize both components of a procedure at once, instead of using the current one-step refinement process.

Also, the procedures are here only combinations of two subtasks, it could be interesting to see if the process can extend to combinations of any number of subtasks.

Finally, proving the potency of our IM-PB learner on a real robotic setup could show its interest for actual robotic application. We are currently designing such an experiment.

References

[1] M. Lungarella, G. Metta, R. Pfeifer, and i. G. Sandin, "Developmental robotics: a survey," *Connection Science*, vol. 15, no. 4, pp. 151–190, 2003.

[2] E. Theodorou, J. Buchli, and S. Schaal, "reinforcement learning of motor skills in high dimensions: a path integral approach," in *robotics and automation (icra), 2010 ieee international conference on*, 2010, pp. 2397–2403. [Online]. Available: http://www-clmc.usc.edu/publications/T/theodorou-ICRA2010.pdf

[3] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *Humanoids*, 2011, pp. 231–238.

[4] E. Deci and R. M. Ryan, *Intrinsic Motivation and self-determination in human behavior*. New York: Plenum Press, 1985.

[5] A. Baranes and P.-Y. Oudeyer, "Intrinsically motivated goal exploration for active motor learning in robots: A case study," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 1766 –1773.

[6] M. Rolf, J. Steil, and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Trans. Autonomous Mental Development*, vol. 2, no. 3, pp. 216–229, 09/2010 2010.

[7] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[8] M. Lopes and P.-Y. Oudeyer, "The Strategic Student Approach for Life-Long Exploration and Learning," in *IEEE Conference on Development and Learning / EpiRob*, San Diego, États-Unis, Nov. 2012. [Online]. Available: http://hal.inria.fr/hal-00755216

[9] Y. Baram, R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," *The Journal of Machine Learning Research,*, vol. 5, pp. 255–291, 2004.

[10] S. M. Nguyen, A. Baranes, and P.-Y. Oudeyer, "Bootstrapping intrinsically motivated learning with human demonstrations," in *IEEE International Conference on Development and Learning*, Frankfurt, Germany, 2011.

[11] N. Duminy, S. M. Nguyen, and D. Duhaut, "Strategic and interactive learning of a hierarchical set of tasks by the Poppy humanoid robot," in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, Sep. 2016, pp. 204–209.

[12] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, pp. 181–211, August 1999. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(99)00052-1

[13] S. Forestier and P.-Y. Oudeyer, "Curiosity-driven development of tool use precursors: a computational model," in *38th Annual Conference of the Cognitive Science Society (CogSci 2016)*, 2016, pp. 1859–1864.

[14] A. G. Barto, G. Konidaris, and C. Vigorito, "Behavioral hierarchy: exploration and representation," in *Computational and Robotic Models of the Hierarchical Organization of Behavior*. Springer, 2013, pp. 13–46. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-39875-9_2

[15] S. M. Nguyen and P.-Y. Oudeyer, "Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner," *Paladyn Journal of Behavioural Robotics*, vol. 3, no. 3, pp. 136–146, 2012. [Online]. Available: http://dx.doi.org/10.2478/s13230-013-0110-z

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 763–768. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5152385