Effects of social guidance on a robot learning sequences of policies in hierarchical learning

Nicolas Duminy
Université Bretagne Sud
Lorient, France
Email: nicolas.duminy@telecom-bretagne.eu

Sao Mai Nguyen
IMT Atlantique, Lab-STICC, UBL
F-29238 Brest, France
Email: nguyensmai@gmail.com

Dominique Duhaut
Université Bretagne Sud
Lorient, France
Email: dominique.duhaut@univ-ubs.fr

***Abstract*—We aim for a robot capable to learn sequences of motor policies to achieve a field of complex tasks. In this paper, we consider a set of interrelated complex tasks hierarchically organized. To address this high-dimensional mapping between a continuous high-dimensional space of tasks and an infinite dimensional space of sequences of policies, we introduce a framework called "procedure", which enables the creation of sequences of policies by combining previously learned skills.**

We propose an active learning algorithmic architecture, capable of organizing its learning process in order to achieve a field of complex tasks by learning sequences of primitive motor policies. Based on heuristics of goal-babbling, social guidance, strategic learning guided by intrinsic motivation, and the "procedure" framework, our algorithm can actively decide on which outcome to focus and which exploration strategy to apply.

We show that a simulation industrial robot can tackle the learning of complex motor policies and adapt this complexity to that of the task at hand. Owing to its exploration strategies, it can discover the levels of difficulty of the tasks, and learn the hierarchy between tasks so as to combine simple tasks to complete a complex task.

Keywords: intrinsic motivation, social guidance, hierarchical learning, curriculum learning, continual learning, active imitation learning.

I. Introduction

Taking a developmental robotic approach [1], we combine the approaches of active motor skill learning of multiple tasks, interactive learning and strategic learning into a new learning algorithm. We show its capability to learn a mapping between a continuous space of hierarchically organized outcomes (sometimes referred to as tasks) and a space of complex motor policies (sometimes referred to as actions), using a physical simulator of a robot arm (see Fig. 1).

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 particular task, limiting their capability for multi-task learning. Intrinsic motivation, which is described in psychology as triggering curiosity in human beings [4], has inspired the design of learning algorithms using competence progress measures. They successfully drive the learner's exploration through goal-babbling [5], [6].

However when the dimension of the outcome space increases, these methods become less efficient [7] due to the curse of dimensionality, or when the reachable space of the robot is small compared to its environment. In those cases, heuristics such as social guidance can help by driving its exploration towards interesting and reachable space fast.

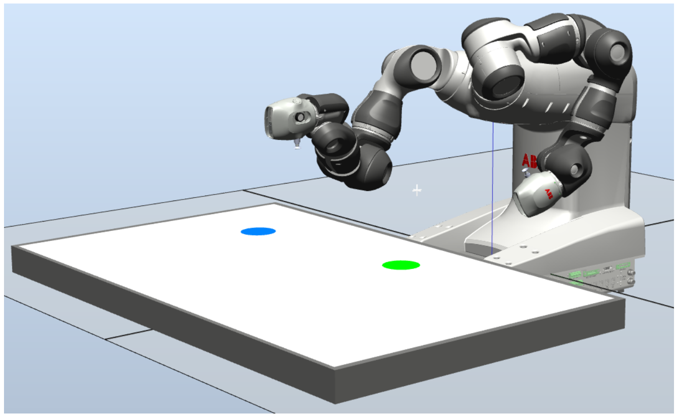
Fig. 1: Experimental setup: The right arm of a Yumi robot can move and touch the interactive table, and move both virtual objects (first object in blue, the second in green). Sound is emitted, depending on the positions of both objects.

B. *Interactive learning*

Combining intrinsically motivated learning and imitation [8] has bootstrapped exploration by providing efficient human demonstrations of motor policies and outcomes.

Also, such a learner proved to be more efficient if requesting actively a human for help when needed instead of being passive, both from the learner and teacher perspectives [9]. This approach is called interactive learning and it enables a learner to benefit from both local exploration and learning from demonstration. Information could be provided to the robot using external reinforcement signals [10], actions [11], advice operators [12], or disambiguation among actions [13]. It enables to include non-robotic experts in the learning process [13]. A key element is to choose when to request human information or learn in autonomy such as to diminish the teacher's attendance.

C. *Strategic learning*

Approaches enabling the learner to choose what to learn (which outcome to focus on) or how to learn (which strategy to use such as imitation) are called strategic learning [14].

They aim at giving an autonomous learner the capability to self-organize its learning process.

Some work has been done to enable a learner to choose which task space to focus on, for instance the SAGG-RIAC algorithm [5] which self-generates goal outcomes. Other approaches enabled a learner to change its strategy [15] and showed it could be more efficient than each strategy alone.

Fewer studies have been made to enable a learner to choose both its strategy and its target outcome. The problem was introduced and studied in [14], and was implemented for an infinite number of outcomes and policies in continuous spaces by the SGIM-ACTS algorithm [16]. This algorithm relies on the empirical evaluation of its learning process to choose actively both which strategy (between autonomous exploration driven by intrinsic motivation and imitations of each of the available human teachers) to use and which outcome to focus on. It showed its potential to learn on a real high dimensional robot a set of hierarchically organized tasks [17]. This is why we consider this approach to learn complex motor policies.

D. 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 [18] 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 also benefit from this task hierarchy (as [19] did for learning tool use with simple primitive policies only), and try to reuse previously acquired skills to build more complex ones. [20] 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 go further in this study: our robot does not know in advance the relationships and dependencies between the tasks. We would like to enable a robot learner to achieve a wide range of tasks that can be inter-related and complex. The learning agent should learn which sequence of policies to use to achieve any task in the infinite field of tasks. It has to face the problem of unlearnability of infinite task and policy spaces, and the curse of dimensionality of sequences of high-dimensionality policy spaces. Thus, the "procedures" framework was introduced in [21] for an intrinsically motivated autonomous learner.

In this paper, we extend [21] by 1/ allowing procedures to be combinations of any number of tasks (and not only two) 2/ studying the effects of interactive learning when the robot can actively request guidance from teachers, on the performance and learning process. Extending the algorithm SGIM-ACTS, we develop a new algorithm called Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB) capable of discovering and using the task hierarchy, along with self-organizing its learning process, to learn a set of complex interrelated tasks using adapted complex policies. First we will formalize and explain our approach, then 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 the field of developmental psychology, we develop a learning algorithm combining interactive learning and autonomous exploration. This learner is driven by intrinsic motivation, can learn and exploit the task hierarchy to build new complex policies through the reuse of known tasks and adapt the complexity of its policies to the task at hand.

In this section, we formalize the problem and our procedures framework, and describe our learning algorithm SGIM-PB.

A. Problem formalization

In our approach, an agent can perform primitive policies π_θ, parametrised by $\theta \in \Pi \subset \mathbb{R}^n$. It can also perform complex motor policies, which are successions of any number of primitive motor policies. The policy space $\Pi^{\mathbb{N}}$ is the combination of subspaces Π^i corresponding to each number of primitives. Those policies induce outcomes in the environment, parametrized by $\omega \in \Omega$. The agent is then to learn the mapping between $\Pi^{\mathbb{N}}$ and Ω: it learns to predict the outcome ω of each policy π (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 are split in outcome subspaces $\Omega_i \subset \Omega$.

We take the trial and error approach and suppose that Ω is a metric space, meaning the learner has a means of evaluating a distance between two outcomes $d(\omega_1, \omega_2)$.

However, to enable our learner to limit the complexity of its policies, the performance metric it is using (1) takes into account the complexity 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).

B. Procedures

Our algorithm is to tackle the learning of hierarchically organized tasks, so learning and exploiting this task hierarchy could ease the learning of the most complex tasks. We defined in [21] procedures in a way that would encourage the learner to reuse previously learned skills and combine them to learn new ones, and extend this definition so as to combine any number of sub-skills. A procedure is henceforth defined as the succession of previously known outcomes $(\omega_1, \omega_2, ..., \omega_n \in \Omega)$ and is noted $\omega_1 \boxplus \omega_2 \boxplus ... \boxplus \omega_n$.

To use a procedure $\omega_1 \boxplus \omega_2 \boxplus ... \boxplus \omega_n$, the learner builds the complex policy π which corresponds to the succession of the policies $\pi_i, i \in [\![1, n]\!]$ (potentially complex as well), where

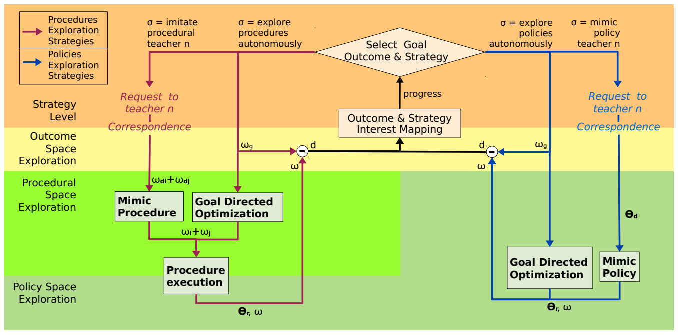
Procedures Exploration Strategies
Policies Exploration Strategies
σ = imitate procedural teacher n
σ = explore procedures autonomously
Select Goal Outcome & Strategy
σ = explore policies autonomously
σ = mimic policy teacher n
Strategy Level
Request to teacher n
Correspondence
progress
Outcome & Strategy Interest Mapping
Request to teacher n
Correspondence
Outcome Space Exploration
ωg
d
ω
d
ωg
ω
Procedural Space Exploration
ωdi+ωdj
Mimic Procedure
Goal Directed Optimization
ωi+ωj
Θd
Procedure execution
Goal Directed Optimization
Mimic Policy
Policy Space Exploration
Θr, ω
Θr, ω

Fig. 2: Architecture of the SGIM-PB algorithm

each π_i reaches ω_i the best, and execute it. A procedure being only a mean to build a complex policy, the learner does not check if any of the subtasks which compose it are actually reached. As the subtasks ω_i may be unknown from the learner, the procedure is modified before execution according to Algo. 1, by replacing them by the closest known tasks ω'_i.

Algorithm 1 Procedure adaptation

```
Input: $(\omega_1, ..., \omega_n) \in \Omega^n$
Input: inverse model $L$
  for $i \in [\![1, n]\!]$ do
    $\omega'_i \leftarrow$Nearest-Neighbour($\omega_i$) // get the nearest outcome known from $\omega_i$
    $\pi'_i \leftarrow L(\omega'_i)$ // get the known complex policy that reached $\omega'_i$
  end for
  return $\pi = \pi'_1 ... \pi'_n$
```

C. Socially Guided Intrinsic Motivation with Procedure Babbling

The SGIM-PB algorithm (see Fig. 2) learns by episodes in which a goal outcome $\omega_g \in \Omega$ and an exploration strategy σ have been selected. In an episode under :

- the autonomous policy space exploration strategy, the learner tries to optimize the policy π_θ to produce ω_g by choosing between random policy exploration and local optimization using the SAGG-RIAC algorithm [5].
- the autonomous procedure exploration strategy, the learner tries to optimize in a procedure space of type $\omega_i \boxplus \omega_j$ to produce ω_g by choosing between random exploration of procedures (which selects two subtasks at random) and local procedure optimization which optimizes a procedure using local linear regression. The procedure is then adapted and used to build the complex policy which is executed according to Algo. 1.
- the mimicry of a policy teacher strategy, the learner requests a policy demonstration π_θ from the chosen teacher, selected by the teacher as the closest from the goal outcome ω_g in its demonstration repertoire. The learner then repeats the demonstrated policy.
- the imitation of a procedure teacher strategy, the learner requests a procedure demonstration of size 2 $\omega_{di} \boxplus \omega_{dj}$ which is built by the chosen teacher according to a preset function which depends on the target outcome ω_g. Then the learner tries to reproduce the demonstrated procedure by adapting it, building and executing its combined complex policy, following Algo. 1.

After each episode, the learner stores the executed procedures, policies and reached outcomes in its episodic memory. It then computes its competence at reaching the goal ω_g, which is the euclidean distance $d(\omega_r, \omega_g)$ between ω_g and the outcome ω_r it actually reached. Then it updates its interest model by computing the interest of the goal outcome and each outcome reached (including the outcome spaces reached but not targeted): $interest(\omega, \sigma) = p(\omega)/K(\sigma)$, where $K(\sigma)$ is the cost of the strategy used and the progress $p(\omega)$ is the derivative of the competence.

The learner then uses these interest measures to partition the outcome space Ω in regions of high and low progress. This process is described in details in [16].

The strategy and goal outcome are chosen stochastically with a density probability proportional to the empirical progress measured in each region R_n of the outcome space Ω, as detailed in [17].

Strategically choosing at each episode between social guidance or autonomous exploration, based on intrinsic motivation, SGIM-PB self-organizes its exploration of the policy, procedure and task spaces, in order to tackle the learning of a set of multiple complex tasks and exploit the task hierarchy.

III. Experiment

We designed an experimental setup, in which the 7 DOF right arm of an industrial Yumi robot by ABB can interact with an interactive table and its virtual objects. It can learn an infinite number of hierarchically organized tasks regrouped in 5 types of tasks, using complex motor policies of unrestricted size.

A. Setup

Fig. 1 shows the robot facing an interactive table. The robot learns to interact with it using the tip of its arm (the tip of the vacuum pump below its hand). The position of the arm's tip on the table is noted (x_0, y_0). Two virtual objects (disks of radius $R = 4cm$) can be picked and placed, by placing the arm's tip on them and moving it at another position on the table. Once interacted with, the final positions of the two objects, respectively (x_1, y_1) and (x_2, y_2). Only one object can be moved at a time, otherwise the setup is blocked and the robot's motion cancelled. If both objects have been moved, a sound is emitted by the interactive table, parametrised by its frequency f, its intensity level l and its rhythm b. The emitted sound depends on the relative position of both objects and the absolute position of the first object. The sound parameters are computed as follow:

$$f = (D/4 - d_{min})4/D \quad (2)$$

$$l = 1 - 2(log(r) - log(r_{min}))/(log(D) - log(r_{min})) \quad (3)$$

$$b = (|\varphi|/\pi) * 0.95 + 0.05 \quad (4)$$

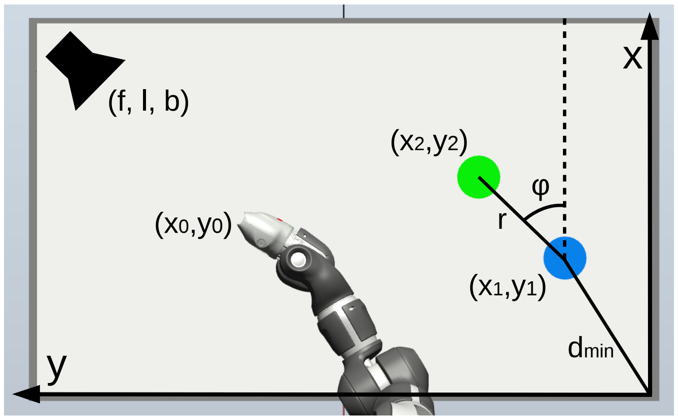

Fig. 3: Representation of the interactive table: the first object is in blue, the second one in green, the produced sound is also represented in top left corner

where D is the diagonal of the interactive table, $r_{min} = 2R$, (r, φ) the polar coordinate of the second object in the system centred on the first one, and d_{min} is the distance between the first object and the closest table corner (see Fig. 3).

The motions of the Yumi robot are executed using a physical simulation (using the Robotstudio software by ABB). The interactive table and its behaviour is simulated and its state is refreshed after each primitive motor policy executed. The robot is not allowed to collide with the interactive table. In this case, the motor policy is cancelled and reaches no outcomes. The arm itself has 7 DOF. Before each attempt, the robot is set to its initial position and the environment is reset.

B. Experiment variables

1) Policy spaces: The motions of each joint are controlled by Dynamic Movement Primitives (DMP). a_i controlling it, parametrised by the end joint angle $g^{(i)}$, and one basis functions for the forcing term, parametrized by its weight $w^{(i)}$. We are using the original form of the DMP from [22] and we keep the same notations. A primitive motor policy is simply the concatenation of those DMP parameters for all joints:

$$\theta = (a_0, a_1, a_2, a_3, a_4, a_5, a_6) \quad (5)$$

$$where\ a_i = (w^{(i)}, g^{(i)}) \quad (6)$$

Two or more primitive policies $(\pi_{\theta_0}, \pi_{\theta_1}, ...)$ can be combined in a complex policy π.

2) Task spaces: The task spaces the robot learns are hierarchically organized:

- $\Omega_0 = \{(x_0, y_0)\}$: the positions touched by the robot on the table;
- $\Omega_1 = \{(x_1, y_1)\}$: the positions where the robot placed the first object on the table;
- $\Omega_2 = \{(x_2, y_2)\}$: the positions where the robot placed the second object on the table;
- $\Omega_3 = \{(x_1, y_1, x_2, y_2)\}$: the positions where the robot placed both objects;
- $\Omega_4 = \{(f, l, b)\}$: the sounds produced by the table;

C. The teachers

To help the SGIM-PB learner, procedural teachers (with a strategical cost $K(\sigma) = 5$) were available for every outcome space except Ω_0. Each teacher is only capable to give procedures according to its outcome space of expertise, knows the task hierarchy and indicate procedures according to a construction rule:

- ProceduralTeacher1 (Ω_1): $\omega_0 \boxplus \omega_0'$ where $\omega_0 \in \Omega_0$ corresponds to the initial position of the first object on the table, and $\omega_0' \in \Omega_0$ to its desired final position;
- ProceduralTeacher2 (Ω_2): $\omega_0 \boxplus \omega_0'$ where $\omega_0 \in \Omega_0$ corresponds to the initial position of the second object on the table, and $\omega_0' \in \Omega_0$ to its desired final position;
- ProceduralTeacher3 (Ω_3): $\omega_1 \boxplus \omega_2$ where $\omega_1 \in \Omega_1$ corresponds to the first object desired final position on the table, and $\omega_2 \in \Omega_2$ to that of the second one;
- ProceduralTeacher4 (Ω_4): $\omega_1 \boxplus \omega_2$, where $\omega_1 \in \Omega_1$ is the final position of the first object, chosen as to both be on the semi-diagonal going from bottom-right corner to the centre of the table and corresponding to the desired sound frequency, and $\omega_2 \in \Omega_2$ is the final position of the second object which relative position to first one corresponds to the desired sound level and rhythm.

We also added different configurations of policy teachers (with a strategical cost of $K(\sigma) = 10$), each expert of one outcome space:

- PolicyTeacher0 (Ω_0): 11 demos of primitive policies;
- PolicyTeacher1 (Ω_1): 10 demos of size 2 policies;
- PolicyTeacher2 (Ω_2): 8 demos of size 2 policies;
- PolicyTeacher34 $(\Omega_3 \times \Omega_4)$: 73 demos of size 4 policies.

D. Evaluation method

To evaluate our algorithm, we created a benchmark linearly distributed across the Ω_i, of 18,800 points. The evaluation consists in computing mean Euclidean distance between each of the benchmark outcomes and their nearest neighbour in the learner dataset. When the learner is incapable to at least reach the outcome space, the evaluation is set to 5. The evaluation is repeated regularly across the learning process.

Then to assess the efficiency of our algorithm, we are comparing the results of the following algorithms:

- RandomPolicy: random exploration of the policy space Π;
- SAGG-RIAC: autonomous exploration of the policy space Π driven by intrinsic motivation;
- SGIM-ACTS: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration of the policy space Π and mimicry of one of the policy teachers;
- SGIM-PB: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration strategies (either of the policy space or the procedural space) and mimicry of one of the available teachers procedural teachers and PolicyTeacher0.

Each algorithm was run once, except for the SGIM-PB which was run 3 times (results averaged on all runs). We

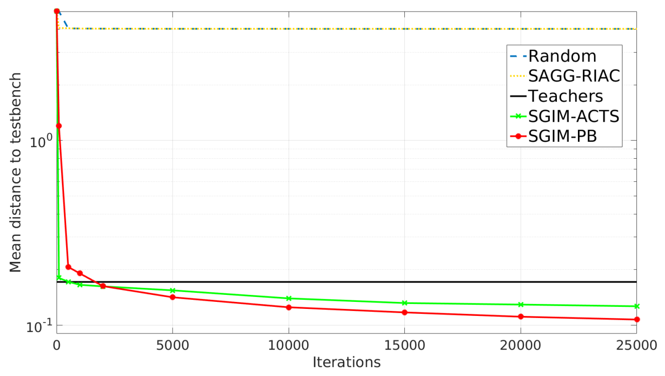

Fig. 4: Evaluation of all algorithms throughout learning process, SGIM-PB has a final standard deviation of 0.00446

also added another result as a threshold corresponding to the evaluation of a learner knowing only the combined skills of every policy teachers for the whole learning process, called Teachers.

IV. Results

The Fig. 4 shows the global evaluation of all tested algorithms, which corresponds to the mean error made by each algorithm to reproduce the benchmarks with respect to the number of complete complex motor policies tried during the learning. We can see that both autonomous learners (Random-Policy and SAGG-RIAC) have higher final levels of error than the others, which shows this setup was tough to learn without demonstrations. We also show that both the SGIM-PB and the SGIM-ACTS learners have errors dropping lower than the Teachers result (in black), showing they went further than the provided policy demonstrations. Also both have about the same final evaluation, SGIM-PB even slightly outperforming SGIM-ACTS, showing that procedural teachers can replace policy teachers for helping learning complex tasks. If we look at the evaluation per outcome space, on Fig. 5, we also see that both autonomous learners were not able to move any of the objects as they did not reach any of the complex outcome spaces $\Omega_1, \Omega_2, \Omega_3, \Omega_4$. Moreover, both SGIM learners have similar final evaluation measures for the $\Omega_0, \Omega_1, \Omega_2$ spaces and SGIM-PB outperforms SGIM-ACTS on the most complex tasks Ω_3, Ω_4. Thus, procedural teachers are well adapted to tackle the most complex and hierarchical outcome spaces.

If we look at the learning process of the SGIM-PB learner, we can see the proportion of strategical choices made by the learner at the beginning of each episode. Fig. 6 shows those choices per outcome space and strategy, and we see that the SGIM-PB learner was capable to organize its learning process. We can see that the learner spent most of its time learning the most complex outcome spaces Ω_3, Ω_4 and especially the highest dimension space Ω_3. Also the learner spent most time using autonomous exploration strategies, which reduces the need for the teachers attendance. We also see that the learner explored mostly the procedural space for the most complex outcome spaces Ω_3, Ω_4, while more relying on policy exploration for the least complex outcome space Ω_0. We can also see that the learner figured on the overall which teacher

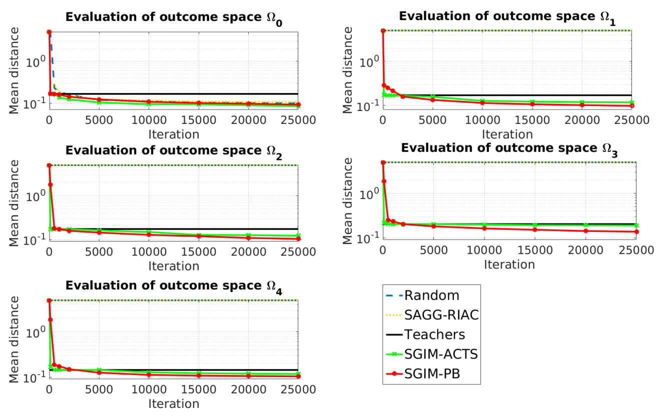

Fig. 5: Evaluation of all algorithms per outcome space (RandomPolicy and SAGG-RIAC are superposed on all evaluations except for Ω_0)

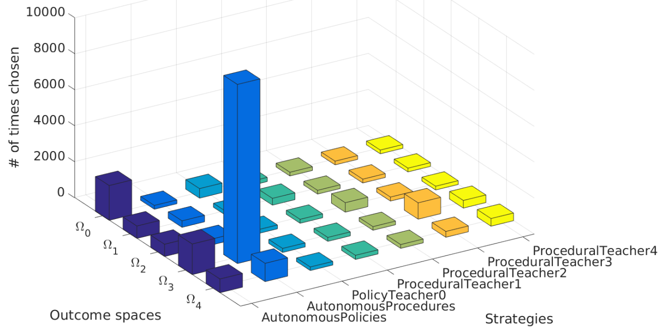

Fig. 6: Choices of strategy and goal outcome for the SGIM-PB learner

was more appropriate for each outcome space. Even though it used almost equally ProceduralTeacher 3 and 4 for the Ω_4 space as those spaces are related.

To see if the SGIM-PB learner was capable to adapt the complexity of its actions to the task at hand, we analysed which policy size would be chosen by the local policy optimization function, for each point of the evaluation testbench. We computed this percentage for three outcome spaces of increasing complexity : Ω_0, Ω_1 and Ω_4. We showed it on Fig. 7. We can see that SGIM-PB is able to limit the size of its policies: using mostly primitive policies and 2-primitive policies for Ω_0, 2-primitive policies for Ω_1, and 4-primitive policies for Ω_4. Although, it could be wondered why the Ω_0 outcome space had been associated with size 2 policies, and not only primitives. This is certainly due to the fact that SGIM-PB set goals in the Ω_0 outcome space far fewer times than on the more complex outcome spaces (2000 times against more than 18,000 times for Ω_3 and Ω_4). So it tried a lot of complex policies which reached Ω_0 as any action that moves any object or makes sound $(\Omega_1, \Omega_2, \Omega_3, \Omega_4)$ also touches the table.

V. Conclusion and Future Work

These results show the capability of SGIM-PB to tackle the learning of a set of multiple interrelated complex tasks using complex motor policies. It showed it was capable to organize its learning process. Although our SGIM-PB learner was unrestricted in the size of policies it could execute, it

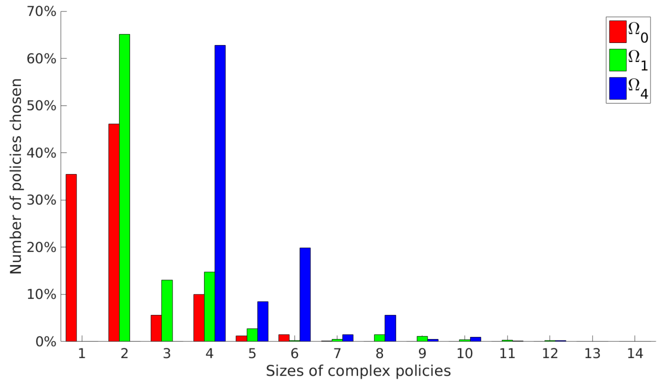

Fig. 7: Percentage of policies chosen per policy size by the SGIM-PB learner for each outcome space

proved it could correctly adapt the complexity of its policies to the task at hand. It showed it could successfully discover the task hierarchy and use it to progress further.

We showed a robotic learner could benefit from demonstrations, and especially identify the most relevant teachers for each outcome subspace. Moreover, our SGIM-PB learner outperforms the SGIM-ACTS learner on the most complex outcome spaces, owing to the procedure framework which enables it to learn and exploit the task hierarchy of this experimental setup and previously learned skills. It also showed that demonstrations of procedures can efficiently replace demonstrations of complex motor policy. They are all the more interesting as they require less robotic knowledge from the teachers such as the kinematics of the robot or the correspondance problems. Instead, the teacher needs only to focus on the knowledge about the tasks and their relationships.

However, this experiment, though using an industrial robot, was conducted on simulation. We are currently adapting the setup for a real Yumi robot. Moreover, we would like to repeat the experiment to make a statistic analysis. Also, even if the procedure framework is defined for unrestrained number of subtasks, we want to design an experimental setup to actually test it, instead of keeping only 2-size procedures.

Acknowledgement

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.

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] J. Peters and S. Schaal, "Natural actor critic," *Neurocomputing*, no. 7-9, pp. 1180–1190, 2008.

[3] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 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*. IEEE, 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] S. M. Nguyen, A. Baranes, and P.-Y. Oudeyer, "Bootstrapping intrinsically motivated learning with human demonstrations," in *IEEE International Conference on Development and Learning*, vol. 2. IEEE, 2011, pp. 1–8.

[9] M. Cakmak, C. Chao, and A. L. Thomaz, "Designing interactions for robot active learners," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 2, pp. 108–118, 2010.

[10] A. L. Thomaz and C. Breazeal, "Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers," *Connection Science*, vol. 20 Special Issue on Social Learning in Embodied Agents, no. 2,3, pp. 91–110, 2008.

[11] D. H. Grollman and O. C. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 261–266.

[12] B. D. Argall, B. Browning, and M. Veloso, "Learning robot motion control with demonstration and advice-operators," in *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, September 2008, pp. 399–404.

[13] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, p. 1, 2009.

[14] M. Lopes and P.-Y. Oudeyer, "The strategic student approach for life-long exploration and learning," in *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1–8.

[15] 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.

[16] 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.

[17] 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.

[18] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[19] 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.

[20] 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.

[21] 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," in *IEEE International Robotics Conference*, 2018.

[22] 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.