*Article*

# Intrinsically Motivated Open-Ended Multi-Task Learning Using Transfer Learning to Discover Task Hierarchy

**Nicolas Duminy [1,2], Sao Mai Nguyen [2,3,*], Junshuai Zhu [2], Dominique Duhaut [1] and Jerome Kerdreux [2]**

[1] Département Mathématiques Informatique Statistique, Université Bretagne Sud,Lab-STICC, UBL, 56321 Lorient, France; nicolas.duminy@telecom-bretagne.eu (N.D.); dominique.duhaut@univ-ubs.fr (D.D.)
[2] IMT Atlantique, Lab-STICC, UBL, 29238 Brest, France; junshuai.zhu@imt-atlantique.net (J.Z.); jerome.kerdreux@imt-atlantique.fr (J.K.)
[3] Flowers Team, U2IS, ENSTA Paris, Institut Polytechnique de Paris & Inria, 91120 Palaiseau, France
* Correspondence: nguyensmai@gmail.com

**Abstract:** In open-ended continuous environments, robots need to learn multiple parameterised control tasks in hierarchical reinforcement learning. We hypothesise that the most complex tasks can be learned more easily by transferring knowledge from simpler tasks, and faster by adapting the complexity of the actions to the task. We propose a task-oriented representation of complex actions, called *procedures*, to learn online task relationships and unbounded sequences of action primitives to control the different observables of the environment. Combining both goal-babbling with imitation learning, and active learning with transfer of knowledge based on intrinsic motivation, our algorithm self-organises its learning process. It chooses at any given time a task to focus on; and what, how, when and from whom to transfer knowledge. We show with a simulation and a real industrial robot arm, in cross-task and cross-learner transfer settings, that task composition is key to tackle highly complex tasks. Task decomposition is also efficiently transferred across different embodied learners and by active imitation, where the robot requests just a small amount of demonstrations and the adequate type of information. The robot learns and exploits task dependencies so as to learn tasks of every complexity.

**Keywords:** curriculum learning; continual learning; hierarchical reinforcement learning; interactive reinforcement learning; imitation learning; multi-task learning; active imitation learning; hierarchical learning; intrinsic motivation

**Citation:** Duminy, N.; Nguyen, S.M.; Zhu, J.; Duhaut, D.; Kerdreux, J. Intrinsically Motivated Open-Ended Multi-Task Learning Using Transfer Learning to Discover Task Hierarchy. *Appl. Sci.* **2021**, *11*, 975. https://doi.org/10.3390/app11030975

## 1. Introduction

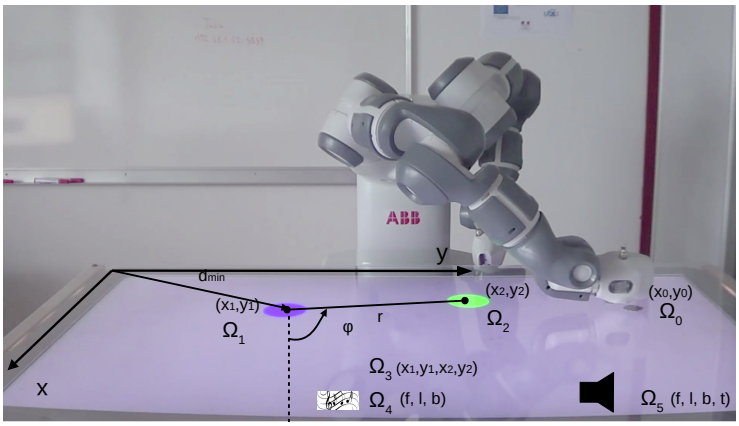



**Figure 1.** Real Yumi setup: the 7-DOF industrial robot arm can produce sounds by moving the blue and green objects and touching the table. See https://youtu.be/6gQn3JGhLvs for an example task.

Let us consider a reinforcement learning (RL) [1] robot placed in an environment surrounded by objects, without external rewards, but with human experts' help. How

can the robot learn multiple tasks such as manipulating an object, at the same time as combining objects together or other complex tasks that requires multiple steps?

In the case of tasks with various complexities and dimensionalities, without a priori domain knowledge, the complexities of actions considered should be unbounded. If we relate the complexity of actions to their dimensionality, actions of unbounded complexity should belong to spaces of unbounded dimensionality. For instance, if an action primitive of dimension $n$ is sufficient for placing an object to a position, a sequence of 2 primitives, i.e., an action of dimension $2n$ is sufficient to place the stick on 2 xylophone keys. Nevertheless, tunes have variable lengths and durations. Likewise, as a conceptual illustration, an unbounded sequence of actions is needed to control the interactive table and play tunes of any length in a setup like in Figure 1. In this work, we consider that actions of unbounded complexity can be expressed as *action primitives* and unbounded *sequences of action primitives*, also named in [2] respectively *micro* and *compound actions*. The agent thus needs to estimate the complexity of the task and deploy actions of the corresponding complexity.

To solve this unbounded problem, the learning agent should be *starting small* before trying to learn more complex tasks as theorised in [3]. Indeed, in multi-task learning problems, some tasks can be compositions of simpler tasks (which we call *'hierarchically organised tasks'*). This approach has been coined *'curriculum learning'* in [4]. The idea of this approach is to use the knowledge acquired from simple tasks to solve more complex tasks or high level of hierarchy tasks, or in other words, to leverage transfer learning (TL) [5–7]. Uncovering the relationship between tasks is useful for transferring knowledge from a task to another. The insight behind TL is that generalization may occur not only within tasks, but also across tasks. This is relevant for compositional tasks. But how can the learning agent discover the decomposition of tasks and the relationship between tasks? Moreover, transfer of knowledge between tasks can also be completed by transfer of knowledge from teachers. Indeed, humans and many animals do not just learn a task by trial and error. Rather, they extract knowledge about how to approach a problem from watching other people performing a similar task. Behavioural psychology studies [8,9] highlight the importance of social and instructed learning, "*including learning about the consequences, sequential structure and hierarchical organisation of actions*" [10]. Imitation is a mechanism for emerging representational capabilities [11]. How can imitation enable the task decomposition into subtasks, and which kind of information should be transferred from the teacher to the learning agent to enable effective hierarchical reinforcement learning? How robust is this transfer to correspondence problems? How can teachers avoid demonstrations that could correspond to behaviours the agent already masters or require pre-requisites the robot has not learned yet?

This work addresses **multi-task learning in open-ended environments by studying the role of transfer of knowledge across tasks** with the hypothesis that some tasks are interrelated, and the role of **transfer of knowledge from other learners or experts** to determine how information is best transferred for hierarchical reinforcement learning: when, what and whom to imitate?

## 2. State of the Art

To learn unbounded sequences of motor actions for multiple tasks, we examine recent methods for curriculum learning based on intrinsic motivation. We also review the methods for hierarchical reinforcement learning and imitation learning, which can be described as two types of transfer of knowledge.

### *2.1. Intrinsic Motivation for Cross-Task Interpolation*

In continual learning in an open-ended world without external rewards, to discover repertoires of skills, agents must be endowed with intrinsic motivation (IM), which is described in psychology as triggering curiosity in human beings [12], to explore the diversity of outcomes it can cause and to control its environment [13,14]. These methods use a reward function that is not shaped to fit a specific task but is general to all tasks

the robot will face. Tending towards life-long learning, this approach, also called artificial curiosity, may be seen as a particular case of reinforcement learning using a reward function parametrised by internal features to the learning agent. One important form of IM system is the ability to autonomously set one's own goals among the multiple tasks to learn. Approaches such as [15,16] have extended the heuristics of IM with goal-oriented exploration, and proven to be able to learn fields of tasks in continuous task and action spaces of high but bounded dimensionality. More recently, IMGEP [17] and CURIOUS [18] have combined intrinsic motivation and goal babbling with deep neural networks and replay mechanisms. They could select goals in a developmental manner from easy to more difficult tasks. Nevertheless, these works did not leverage cross-goal learning but only used interpolation between parametrised goals over a common memory dataset.

### *2.2. Hierarchically Organised Tasks*

Nevertheless, in the case of tasks with various complexities and dimensionalities, especially with action spaces of unbounded dimensionality, those methods become intractable and the volume of the task and action spaces to explore grows exponentially. In that case, exploiting the relationships between the tasks can enable a learning agent to tackle increasingly complex tasks more easily, and heuristics such as social guidance can highlight these relationships. The idea would be to treat complex skills as assembly tasks, i.e., sequences of simpler tasks. This approach is in line with descriptions of motor behaviour of humans and primates as composing their early motions and being recombined after a maturation phase into sequences of action primitives [19]. In artificial systems, this idea has been implemented as a neuro-dynamic model by composing action primitives in [20] and has been proposed in [21] to learn subtask policies and a scheduler to switch between subtasks, with offline off-policy learning, to derive a solution that is time-dependent on a scheduler. On the other hand, options were proposed as a temporally abstract representation of complex actions made of lower-level actions and revealed faster to reach interesting subspaces as reviewed in [22]. Learning simple skills then combining them by skill chaining is shown in [23] more effective than learning the sequences directly. Other approaches using temporal abstraction and hierarchical organization have been proposed [24].

More recently, Intrinsic Motivation has also tacked hierarchical RL to build increasingly complex skills by discovering and exploiting the task hierarchy using planning methods [25]. However it does not model explicitly a representation of the task hierarchy, letting planning compose the sequences in the exploitation phase. IM has also been used with temporal abstraction and deep leaning in h-DQN [26] with meta-level learning subgoals and controller level policies over atomic actions. However h-DQN was only applied to discrete state and action spaces. A similar idea has been proposed for continuous action and state spaces in [27], where the algorithm IM-PB relies on a representation, called *procedure*, of the task decomposition. A fully autonomous intrinsically motivated learner successfully discovers and exploits the task hierarchy of its complex environment, while still building sequences of action primitives of adapted sizes. These approaches could generalise across tasks by re-using the policies of subgoals for more complex tasks, once the task decomposition is learned. We would like to investigate here for continuous action and state spaces, the role transfer of knowledge on task decomposition in hierarchical reinforcement learning, when there are several levels of hierarchy; and how transfer learning operates when tasks are hierarchically related compared to when tasks are similar.

### *2.3. Active Imitation Learning (Social Guidance)*

Imitation learning techniques or Learning from Demonstrations (LfD) [28,29] provide human knowledge for complex control tasks. However, imitation learning is often limited by the set of demonstrations. To overcome sub-optimality and noisiness of demonstrations, imitation learning has recently been combined with RL exploration, for instance when initial human demonstrations have successfully initiated RL in [30,31]. In [32] combination

of transfer learning, learning from demonstration and reinforcement learning significantly improve both learning time and policy performance for a single task. However, in the previously cited works, two hypotheses are frequently made:

- the transfer of knowledge needs only one type of information. However, for multi-task learning, the demonstrations set should provide different types of information depending on the task and the knowledge of the learning. Indeed, in imitation learning works, different kinds of information for transfer of knowledge have been examined separately, depending on the setup at hand: external reinforcement signals [33], demonstrations of actions [34], demonstrations of procedures [35], advice operators [36] or disambiguation among actions [37]. The combination of different types of demonstrations has been studied in [38] for multi-task learning, where the proposed algorithm Socially Guided Intrinsic Motivation with Active Choice of Teacher and Strategy (SGIM-ACTS) showed that imitating a demonstrated action and outcome has different effects depending on the task, and that the combination of different types of demonstrations with autonomous exploration bootstraps the learning of multiple tasks. For hierarchical RL, algorithm Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB) in [35] could also take advantage of demonstrations of actions and task decomposition. We propose in this study to examine the role of each kind of demonstrations with respect to the control tasks in order to learn task hierarchy.
- the timing of these demonstrations has no influence. However, in curriculum learning the timing of knowledge transfer should be essential. Furthermore, the agent best knows when and what information it needs from the teachers, and active requests for knowledge transfer should be more efficient. For instance, a reinforcement learner choosing when to request social guidance was shown in [39] making more progress. Such techniques are called active imitation learning or interactive learning, and echo the psychological descriptions of infants' selectivity in social partners and its link to their motivation to learn [40,41]. Active imitation learning has been implemented [42] where the agent learns when to imitate using intrinsic motivation for a hierarchical RL problem in a discrete setting. For continuous action, state and goal spaces, the SGIM-ACTS algorithm [38] uses intrinsic motivation to choose not only the kind of demonstrations, but also when to request for demonstrations and who to ask among several teachers. SGIM-ACTS was extended for hierarchical reinforcement learning with the algorithm Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB) in [35]. In this article, we will study whether a transfer of a batch of data or an active learner is more efficient to learn task hierarchy.

### *2.4. Summary: Our Contribution*

We combine both types of transfer of knowledge, across tasks and from experts, to address multi-task learning in a hierarchical RL problem in a non-rewarding, continuous and unbounded environment, where experts with their own field of expertise, unknown to the robot, are available at the learner's request. We propose to continue the approach initiated in [35,43] with the SGIM-PB algorithm that **combines intrinsic motivation, imitation learning and transfer learning** to enable a robot to learn its curriculum by:

- Discovering and exploiting the task hierarchy using a dual representation of complex actions in action and outcome spaces;
- Combining autonomous exploration of the task decompositions with imitation of the available teachers, using demonstrations as task dependencies;
- Using intrinsic motivation, and more precisely its empirical measures of progress, as its guidance mechanism to decide which information to transfer across tasks; and for imitation, when how and from which source of information to transfer.

In this article, we examine how task decomposition can be learned and transferred from a teacher or another learner using the mechanisms of intrinsic motivation in autonomous exploration and active imitation learning for discovering task hierarchy for

cross-task and cross-learner transfer learning. More precisely, while in [35] we showed on a toy simulation faster learning and better precision in the control, in this article, we show on an industrial robot that **task decomposition is pivotal to completing tasks of higher complexity** (by adding more levels of hierarchy in the experimental setup), and we test the properties of our **active imitation of task decomposition : it is valid for cross-learner transfer even in the case of different embodiments, and active imitation proves more efficient than imitation of a batch dataset given from initialisation**. This use-case study enables deeper analysis into the mechanisms of the transfer of knowledge.

The article is organized as follows: we describe our approach in Section 3; and present our setups on the physical simulator and the real robot of an industrial robot arm in Section 4. The results are analysed in Section 5 and discussed in Section 6; finally we conclude this article in Section 7.

## 3. Our Approach

Grounding our work in cognitive developmental robotics [44,45], we propose an intrinsically motivated learner able to self-organize its learning process for multi-task learning of hierarchically organized tasks by exploring action, task and task decomposition spaces. Our proposed algorithm combines autonomous exploration with active imitation learning into a learner discovering the task hierarchy to reuse its previously gained skills for tackling more complex ones, while adapting the complexity of its actions to the complexity of the task at hand.

In this section, we first formalize the learning problem we are facing. Then we describe the algorithm Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB). This algorithm uses a task-oriented representation of task decomposition called procedures to build more and more complex actions, adapting to the difficulty of the task.

### *3.1. Problem Formalization*

Let us consider a robot, able to perform motions through the use of *action primitives* $\pi^\theta \in \Pi$. We suppose that the action primitives are parametrised functions with parameters of dimension $n$. We note the parameters $\theta \in \mathbb{R}^n$. The action primitives represent the smallest unit of motions available to the robot. The robot can also chain multiple action primitives together to form *sequences of action primitives of any size* $k \in \mathbb{N}$. We consider that the robot can execute actions in the form of a sequence of one or several action primitives. We note $\pi$ an action and will precise the parameter $\theta$ in the case of an action primitive $\pi^\theta$. We note $\Pi^{\mathbb{N}}$ the complete space of actions of any size available to the learner.

The environment can change as a consequence of the motions of the robot. We call outcomes $\omega \in \Omega$ these consequences. They can be of various types and dimensionalities, and are therefore split in outcome subspaces $\Omega_i \subset \Omega$. Those outcomes can also be of different complexities, meaning that the actions generating these outcomes may require different numbers of action primitives to chain. The robot aims for learning generalisation (how to reach a range of outcomes as broad as possible), and learning speed. It learns which action to perform depending on the outcomes to generate, known as the inverse model $M : \omega \mapsto \pi$. A *task* is thus a desired outcome, and the inverse model indicates which action can reach it. As more than one action can lead to the same outcome, M is not a function.

We take the trial and error approach, and we suppose that the error can be evaluated and $\Omega$ is a metric space, which means the learner can evaluate a distance between two outcomes $d(\omega_1, \omega_2)$.

### *3.2. Procedures*

Let us note $\mathcal{H}$ the hierarchy of the tasks used by our robot. $\mathcal{H}$ is formally defined as a directed graph where each node is a task $T$. Figure 2 shows a representation of task hierarchy. As our algorithm is tackling the learning of complex hierarchically organized tasks, exploring and exploiting this hierarchy could ease the learning of the most complex tasks.

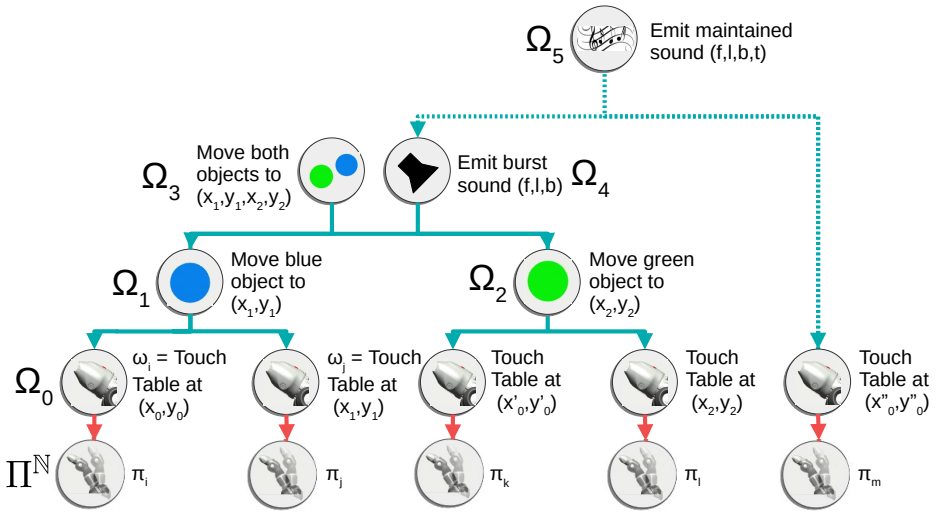



**Figure 2.** Task hierarchy of the Yumi experimental setup : blue lines represent task decomposition into procedures for the simulation setup, dashed lines for the physical setup, red lines for the direct inverse model. Eg. to move a blue object placed initially at $(x_0, y_0)$ to a desired position $(x_1, y_1)$, the robot can carry out task $(\omega_i)$ that consists in moving its end-effector to $(x_0, y_0)$ to pick it, then task $(\omega_j)$ that consists in moving the end-effector to $(x_1, y_1)$. These subtasks are executed with respectively action primitives $\pi^i$ and $\pi^j$. Therefore to move the object, the learning agent can use the sequence of action primitives $(\pi^i, \pi^j)$.

To represent and learn this task hierarchy, we are using the procedure framework. This representation has been created as a way to push the learner to combine previously learned tasks to form more complex ones. A procedure is a combination of outcomes $(\omega_1, \omega_2) \in \Omega^2$. Carrying out a procedure $(\omega_1, \omega_2)$ means executing sequentially each component $\pi_{1,2}$ of the action sequence, where $\pi_i$ is an action that reaches $\omega_i$ $\forall i \in [\![1, 2]\!]$.

We can formalise task decomposition as a mapping from a desired outcome $\omega_g$ to a procedure $P : \omega_g \mapsto (\omega_1, \omega_2)$. $\omega_1$ and $\omega_2$ are then called *subtasks*. In the task hierarchy $\mathcal{H}$, an outcome represents a task node in the graph, while the task decomposition represents the directed edges and the procedure is the list of its successors. $\mathcal{H}$ is initialised as a densely connected graph, and the exploration of SGIM-PB prunes the connections by testing which procedures or task decompositions respect the ground truth. The procedure space or task decomposition space $\Omega^2$ is a new space to be explored by the learner to discover and exploit task decompositions. Our proposition is to derive the inverse model $M$ by using recursively the inverse model $M$ and the task decomposition $P$ until we derive a sequence of action primitives following this recursion:

$$M(\omega_g) \mapsto \begin{cases} (\pi^{\theta_1}, ...\pi^{\theta_k}) & \text{with } k \in \mathbb{N} \\ (M(\omega_1), M(\omega_2)) & \text{if } P(\omega_g) \mapsto (\omega_1, \omega_2) \end{cases}$$

### *3.3. Algorithm*

We here present the algorithm SGIM-PB (Socially Guided Intrinsic Motivation with Procedure Babbling). In Section 5.5, we will also look at the variant named SGIM-TL (Socially Guided Intrinsic Motivation with Transferred Lump), which is provided at initialisation with a dataset of transferred procedures and their corresponding reached outcomes: $\{(\omega_i, \omega_j), \omega_r\}$. The main differences between SGIM-PB and SGIM-TL are outlined and they are contrasted with former versions IM-PB and SGIM-ACTS in Table 1. SGIM-PB and SGIM-TL propose to learn both $M$ and $P$ simultaneously.

**Table 1.** Differences between SGIM-PB, SGIM-TL, SGIM-ACTS and IM-PB.

| Algorithm | Action Representation | Strategies $\sigma$ | Transferred Dataset | Timing of the Transfer | Transfer of Knowledge |
|---|---|---|---|---|---|
| IM-PB [27] | parametrised actions, procedures | auton. action space explo., auton. procedural space explo. | None | NA | cross-task transfer |
| SGIM-ACTS [38] | parametrised actions | auton. action space explo., mimicry of an action teacher | Teacher demo. of actions | Active request by the learner to the teacher | imitation |
| SGIM-TL | parametrised actions, procedures | auton. action space explo., auton. procedural space explo., mimicry of an action teacher, mimicry of a procedure teacher | Another robot's procedures, Teacher demo. of actions and procedures | Procedures transf. at initalization time, Active request by the learner to the teacher | cross-task transfer, imitation |
| SGIM-PB | parametrised actions, procedures | auton. action space explo., auton. procedural space explo., mimicry of an action teacher, mimicry of a procedure teacher | Teacher demo. of actions and procedures | Active request by the learner to the teacher | cross-task transfer, imitation |

SGIM-PB starts learning from scratch. It is only provided with :

- Dimensionality and boundaries of the action primitive space $\Pi$;
- Dimensionality and boundaries of each of the outcome subspaces $\Omega_i \subset \Omega$;
- Dimensionality and boundaries of the procedural spaces $(\Omega_i, \Omega_j) \subset \Omega^2$, defined as all possible pairs of two outcome subspaces.

Our SGIM-PB agent is to collect data in order to learn how to reach all outcomes by generalising from the sampled data. This means it has to learn for all reachable outcomes, the actions or procedures to use to reach the outcome. This corresponds to learning the inverse model $M$. The model uses a local regression based on the k-nearest neighbours from the data collected by exploration. In order to do that, the learning agent is provided with different exploration *strategies* (see Section 3.3.1), that are defined as methods to generate a procedure or action for any given outcome. The 4 types of strategies available to the robot are: two autonomous exploration strategies and two interactive strategies per task type. For the autonomous exploration strategies, we consider action space exploration and procedural space exploration. For the interactive strategies, we consider mimicry of an action or a procedure teacher of the task type: the former's demonstrations are motor actions, while the latter's are procedures.

As these strategies could be more appropriate for some tasks than others, and as their effectiveness can depend on the maturity of the learning process, our learning agent needs to map the outcome subspaces and regions to the best suited strategies to learn them. Thus, SGIM-PB uses an *Interest Mapping* (see Section 3.3.2) that associates to each strategy and region of an outcome space partition an *interest* measure, so as to guide the exploration by indicating which tasks are the most interesting to explore at the current learning stage, and which strategy is the most efficient.

The SGIM-PB algorithm (see Algorithm 1, Figure 3) learns by episodes, and starts each episode by selecting an outcome $\omega_g \in \Omega$ to focus on and an exploration strategy $\sigma$. The strategy and outcome region are selected based on the Interest Mapping by roulette wheel selection also called fitness proportionate selection, where the interest measure serves as fitness (see Section 3.3.2).

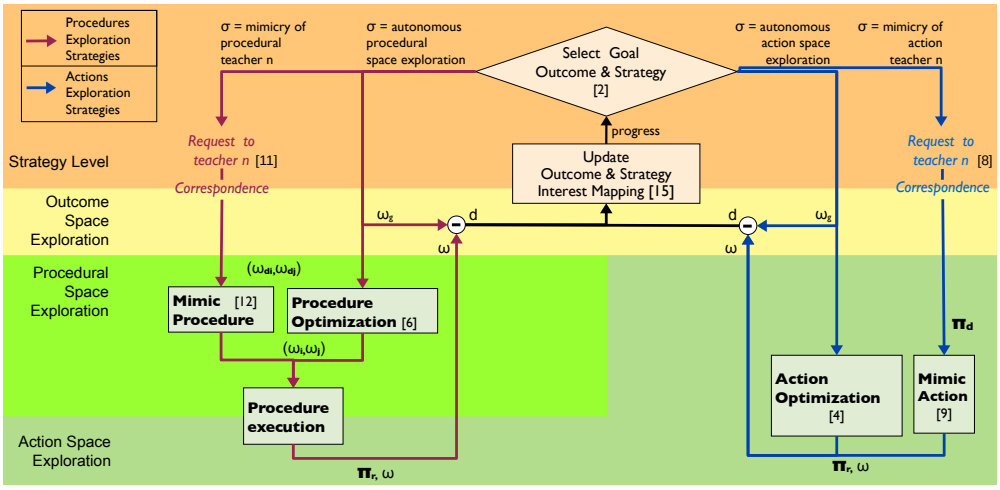



**Figure 3.** SGIM-PB architecture: the arrows show the data transfer between the different blocks, the numbers between brackets refer to specific line number in Algorithm 1.

**Algorithm 1** SGIM-PB and its SGIM-TL variant

**Input:** the different strategies $\sigma_1, ..., \sigma_m$
**Initialization:** partition of outcome spaces $R \leftarrow \bigsqcup_i \{\Omega_i\}$
**Input:** CASE SGIM-TL: transfer dataset $D_3 = \{(\omega_i, \omega_j), \omega_r\}$
**Initialization:** CASE SGIM-TL: episodic memory $Memo \leftarrow \{(\omega_i, \omega_j), \omega_r\}$

**loop**
  $\omega_g, \sigma \leftarrow$ Select Goal Outcome and Strategy($R$)
  **if** $\sigma$ = Autonomous action space exploration strategy **then**
    $Memo \leftarrow$ Goal-Directed Action Optimization($\omega_g$)
  **else if** $\sigma$ = Autonomous procedural space exploration strategy **then**
    $Memo \leftarrow$ Goal-Directed Procedure Optimization($\omega_g$)
  **else if** $\sigma$ = Mimicry of action teacher $i$ strategy **then**
    $(\pi_d, \omega_d) \leftarrow$ ask and observe demonstrated action to teacher $i$
    $Memo \leftarrow$ Mimic Action($\pi_d$)
  **else if** $\sigma$ = Mimicry of procedural teacher $i$ strategy **then**
    $((\omega_{di}, \omega_{dj}), \omega_d) \leftarrow$ ask and observe demonstrated procedure to teacher $i$
    $Memo \leftarrow$ Mimic Procedure($(\omega_{di}, \omega_{dj})$)
  **end if**
  Update $M$ with collected data $Memo$
  $R \leftarrow$ Update Outcome and Strategy Interest Mapping($R$,$Memo$,$\omega_g$)
**end loop**

At each episode, the robot and the environment are reset to their initial states. The learner uses the chosen strategy to build an action (based or not on a procedure), decomposed into action primitives which are then executed sequentially without getting back to its initial position. Whole actions are recorded, along with their outcomes. Each step (after each action primitive execution) of the sequence of action primitives execution is also recorded in *Memo*.

#### 3.3.1. Exploration Strategies

In an episode under the autonomous action space exploration strategy, the learner tries to optimize the action $\pi$ to produce $\omega_g$ by stochastically choosing between random exploration of actions and local regression on the k-nearest action sequence neighbours. The

probability of choosing local regression over random actions is proportional to the distance of $\omega_g$ to its nearest action neighbours. This action optimization is called Goal-Directed Action Optimization and based on the SAGG-RIAC algorithm [15].

In an episode under the autonomous procedural space exploration strategy, the learner tries to optimize the procedure $(\omega_i, \omega_j) \in \Omega^2$ to produce $\omega_g$, by stochastically choosing between random exploration of the procedural space and local regression on the k-nearest procedure neighbours. The probability of choosing local regression over random actions is proportional to the distance of $\omega_g$ to its nearest procedure neighbours. This process is called Goal-Directed Procedure Optimization.

In an episode under the mimicry of an action teacher strategy, the learner requests a demonstration of an action to reach $\omega_g$ from the chosen teacher. The teacher selects the demonstration $\pi_d$ as the action in its demonstration repertoire reaching the closest outcome from $\omega_g$. It has direct access to the parameters of $\pi_d = (\pi_{d1}, \pi_{d2}, \ldots, \pi_{dl})$, and explores locally the action parameters space (we do not consider the correspondence problem from teachers' demonstrations).

In an episode under the mimicry of a procedural teacher strategy, the learner requests a task decomposition of $\omega_g$ from the chosen teacher. The demonstrated procedure $(\omega_{di}, \omega_{dj})$ will define a locality in the procedure space for it to explore.

When performing nearest neighbour searches during the execution of autonomous action and procedure exploration strategies (for local optimization of procedures or actions or when executing a procedure), the algorithm uses a performance metric which takes into account the complexity of the underlying action selected:

$$perf(\omega_g) = d(\omega, \omega_g)\gamma^n \tag{1}$$

where $d(\omega, \omega_g)$ is the normalized Euclidean distance between the target and the reached outcomes $\omega_g$ and $\omega$, $n$ is the size of the action chosen (the length of the sequence of primitives), $\gamma > 1$ is a constant used to balance accuracy and complexity of the action. In our case, we want the learner to mainly focus on accuracy while only avoiding overcomplex actions whenever possible, so we manually set a small $\gamma = 1.2$ very close to the lower limit.

#### 3.3.2. Interest Mapping

After each episode, the learner stores the attempted procedures and actions, along with their reached outcomes in its memory. It then computes its competence in reaching the goal outcome $\omega_g$ by computing the distance $d(\omega_r, \omega_g)$ with the outcome $\omega_r$ it actually reached (if it has not reached any outcome in $\Omega_i$, we use a default value $d_{thres}$). Then interest measure is computed for the goal outcome and all outcomes reached during the episode (including the outcomes from a different subspace than the goal outcome):

$$interest(\omega, \sigma) = p(\omega)/K(\sigma) \tag{2}$$

where the progress $p(\omega)$ is the difference between the best competence for $\omega$ before and after the episode, $K(\sigma)$ is the cost associated to each strategy. $K(\sigma)$ is a meta parameter to favour some strategies such as autonomous ones, to push the robot to rely on itself as much as possible instead of bothering teachers.

The interest measures are then used to partition the outcome space $\Omega$. The trajectory of the episode is added to their partition with hindsight experience replay (both goal and reached outcomes are taken into account), storing the values of the strategy $\sigma$, the outcome parameter, and the interest measure. When the number of outcomes added to a region exceeds a fixed limit, the region is split into two regions with a clustering boundary that separates outcomes with low from those with high interest. This method is explained in more details in [46]. The interest mapping is a tool to identify the zone of proximal development where the interest is maximal, and to organize the learning process.

This interest mapping is initialized with the partition composed of each outcome type $\Omega_i \subset \Omega$. For the first episode, the learning agent always starts by choosing a goal and a strategy at random.

In particular, this interest mapping enables SGIM-PB to uncover the task hierarchy by associating goal tasks and procedures. When testing a specific procedure $(\omega_i, \omega_j) \in \Omega^2$ that produces $\omega$ instead of the goal $\omega_g$ under the procedural space exploration or the mimicry of a procedure teacher strategies, SGIM-PB assesses the performance of this task decomposition and records the trajectory of the episode in the memory. This task decomposition is likely to be reused again during local regression of k-nearest neighbours for tasks close to $\omega$ and $\omega_g$ and for short sequences of primitives, i.e., if its performance $perf(\omega_g) = d(\omega, \omega_g)\gamma^n$ is high. Thus the different task decompositions are compared both for their precision and cost in terms of complexity. At the same time, SGIM-PB updates the interest map for that strategy. If the procedure is not relevant for the goal $\omega_g$, the procedure is ignored henceforward. On the contrary if the procedure is the right task decomposition, the interest $interest(\omega, \sigma)$ for this procedure exploration/mimicry strategy $\sigma$ increases. Thus, conversely, SGIM-PB continues to explore using the same strategy, and tests more procedures for the same region of outcomes. As the number of procedures explored increases and are selected by intrinsic motivation (using the performance and interest criteria), SGIM-PB associates goal tasks to the relevant procedures, hence builds up the adequate task decomposition. These associations to procedures constitute the task hierarchy uncovered by the robot.

## 4. Experiment

In this section, we present the experiment we conducted to assess the capability of our SGIM-PB algorithm. The experimental setup features the 7 DOF right arm of an industrial Yumi robot from ABB which can interact with an interactive table and its virtual objects. Figure 1 shows the robot facing a tangible RFID interactive table from [47]. The robot learns to interact with it using its end-effector. It can learn an infinite number of hierarchically organized tasks regrouped in 5 types of tasks, using sequences of motor actions of unrestricted size. The experiments have been carried out with the physical industrial robot ABB and with simulation software provided with the robot, Robotstudio. While the software provided by the robot can provide static inverse kinematics, it can not provide movement trajectories.

We made preliminary tests of our SGIM-PB learner on a physical simulator of the robot. We will call this the *simulation setup*. In the simulation setup, the end-effector is the tip of the vacuum pump below its hand. The simulation setup will be modified as a setup called *left-arm setup* in Section 5.5.

We also implemented this setup for a physical experiment, to compare both interactive strategies more fairly using SGIM-ACTS and SGIM-PB. For that, we modified the procedural teacher strategy so they have the same limited repertoire from which to draw demonstrations as action teachers. We also added an extra more complex task without demonstrations to compare the autonomous exploration capability of both algorithms. The setup, shown on Figure 1, will be referred to as the *physical setup*. In the physical setup, the end-effector is the bottom part of the hand.

In the following subsections, we describe in more details the physical setup and the simulation setup and their variables, while mentioning their differences. Then we end this section by presenting the teachers and evaluation methods.

### 4.1. Experimental Setup

The position of the arm's tip on the table (see Figure 4) is noted $(x_0, y_0)$. Two virtual objects (disks of radius $R = 4$ cm) 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 are given to the robot by the table, 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 burst sound is emitted by the interactive table, parametrised by its frequency $f$, its intensity level $l$ and its rhythm $b$. It can be maintained for a duration $t$ by touching a new position on the table afterwards. The sound parameters are computed with arbitrary rules so as to have both linear and non linear relationships, as follow:

$$f = (D/4 - d_{min})4/D \tag{3}$$
$$l = 1 - 2(ln(r) - ln(r_{min}))/(ln(D) - ln(r_{min})) \tag{4}$$
$$b = (|\varphi|/\pi) * 0.95 + 0.05 \tag{5}$$
$$t = d_2/D \tag{6}$$

where $D$ is the interactive table diagonal, $(r, \varphi)$ are the polar coordinates of the green object in the system centred on the blue object, $r_{min} = 2R$, $d_{min}$ is the distance between the blue object and closest table corner, and $d_2$ is the distance between the end effector position on the table and the green object .

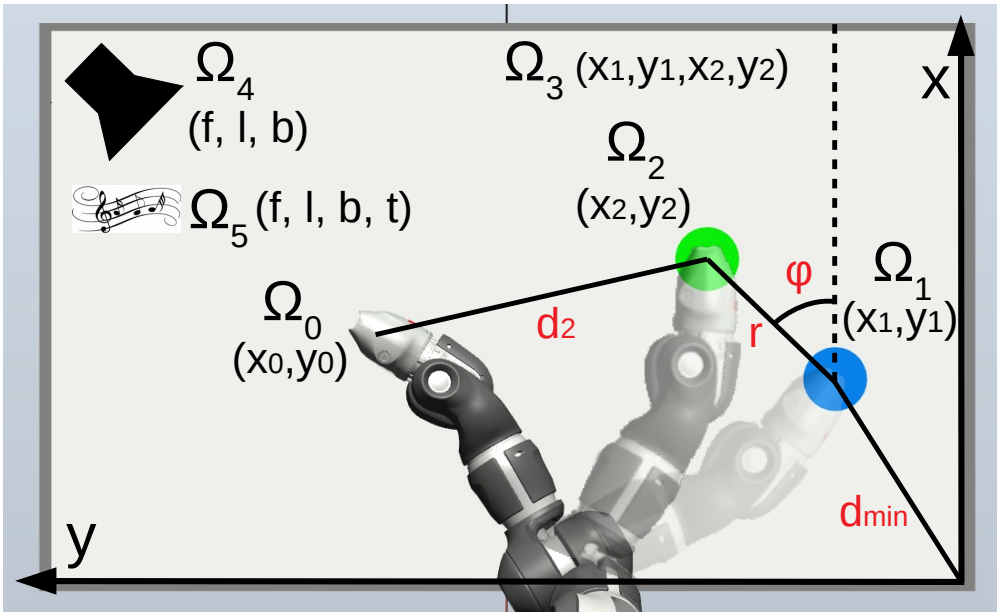



**Figure 4.** Representation of the interactive table: the first object is in blue, the second one in green, the produced burst sound and maintained sound are also represented in top left corner. The outcome spaces are also represented on this figure with their parameters in black, while the sound parameters are in red.

The interactive table state is refreshed after each action primitive executed. The robot is not allowed to collide with the interactive table. In this case, the motor action is cancelled and reaches no outcomes. Before each attempt, the robot is set to its initial position and the environment is reset.

### *4.2. Experiment Variables*

#### 4.2.1. Action Spaces

The motions of each joint are controlled by a one-dimensional Dynamic Movement Primitive (DMP) [48]. We will use the same notations for the DMP parameters, defined by the system:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \tag{7}$$
$$\tau \dot{x} = v \tag{8}$$
$$\tau \dot{s} = -\alpha s \tag{9}$$

where $x$ and $v$ are the position and velocity of the system; $s$ is the phase of the motion; $x_0$ and $g$ are the starting and end position of the motion; $\tau$ is a factor used to temporally scale the system (set to fix the duration of an action primitive execution); $K$ and $D$ are the spring constant and damping term fixed for the whole experiment; $\alpha$ is also a constant fixed for the experiment; and $f$ is a non-linear term used to shape the trajectory called the forcing term. This forcing term is defined as:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{10}$$

where $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ with centers $c_i$ and widths $h_i$ fixed for all primitives. There are 1 weights $w_i$ per DMP, which is therefore simply noted $w$.

The weights of the forcing term and the end positions are the only parameters of the DMP used by the robot. One weight per DMP is used and each DMP controls one joint. The starting position of a primitive is set by either the initial position of the robot (if it is starting a new action) or the end position of the preceding primitive. Therefore an action primitive $\pi^\theta$ is parametrized by:

$$\theta = (a_0, a_1, a_2, a_3, a_4, a_5, a_6) \tag{11}$$

where $a_i = (w^{(i)}, g^{(i)})$ corresponds to DMP parameters of the joint $i$: the final joint angle $g^{(i)}$, and the parameters $w^{(i)}$ of a basis function for the forcing term. The action primitive space is thus $\Pi = \mathbb{R}^{14}$, and the complete action space $(\mathbb{R}^{14})^{\mathbb{N}}$.

#### 4.2.2. Outcome Spaces

The outcome spaces the robot learns are hierarchically organized:

- $\Omega_0 = \{(x_0, y_0)\}$: positions touched on the table;
- $\Omega_1 = \{(x_1, y_1)\}$: positions of the first object;
- $\Omega_2 = \{(x_2, y_2)\}$: positions of the second object;
- $\Omega_3 = \{(x_1, y_1, x_2, y_2)\}$: positions of both objects;
- $\Omega_4 = \{(f, l, b)\}$: burst sounds produced;
- $\Omega_5 = \{(f, l, b, t)\}$: maintained sounds produced.

The outcome space is a composite and continuous space (for the physical setup $\Omega = \bigcup_{i=0}^{5} \Omega_i$, for the simulation setup $\Omega = \bigcup_{i=0}^{4} \Omega_i$), containing subspaces of 2 to 4 dimensions. Multiple interdependencies are present between tasks: controlling the position of either the blue object ($\Omega_1$) or the green object ($\Omega_2$) comes after being able to touch the table at a given position ($\Omega_0$); moving both objects ($\Omega_3$) or making a sound ($\Omega_4$) comes after being able to move the blue and the green object, the maintained sound ($\Omega_5$) is the most complex task of the physical setup. This hierarchy is shown on Figure 2.

Our intuition is that a learning agent should start by making a good progress in the easiest task $\Omega_0$, then $\Omega_1$, $\Omega_2$. Once it mastered those easy tasks, it can reuse that knowledge to learn to achieve the most complex tasks $\Omega_3$ and $\Omega_4$. We will particularly focus on the learning of the $\Omega_4$ outcome space and the use of the procedure framework for it. Indeed in this setting, the relationship between a goal outcome in $\Omega_4$ and the necessary positions of both objects $(\Omega_1, \Omega_2)$ to reach that goal are not linear. So with this setting, we test if the robot can **learn a non-linear mapping between a complex task and a procedural space**. Finally for the physical version, we see if the robot can reach and explore the most complex task $\Omega_5$ in the absence of an allocated teacher.

### *4.3. Teachers*

To help SGIM-PB in the simulation setup, procedural teachers (strategical cost $K(\sigma) = 5$, compared to $K(\sigma) = 1$ for autonomous strategies) were available for every outcome space except $\Omega_0$. Each teacher gives on the fly a procedure adapted to the learner's request, according to its domain of expertise and according to a construction rule:

- ProceduralTeacher1: $\Omega_1 \rightarrow (\Omega_0^2)$;
- ProceduralTeacher2: $\Omega_2 \rightarrow (\Omega_0^2)$;
- ProceduralTeacher3: $\Omega_3 \rightarrow (\Omega_1, \Omega_2)$;
- ProceduralTeacher4: $\Omega_4 \rightarrow (\Omega_1, \Omega_2)$.

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

- ActionTeacher0 ($\Omega_0$): 11 demos of action primitives;
- ActionTeacher1 ($\Omega_1$): 10 demos of size 2 actions;
- ActionTeacher2 ($\Omega_2$): 8 demos of size 2 actions;
- ActionTeacher34 ($\Omega_3$ and $\Omega_4$): 73 demos of size 4 actions.

In the physical setup, we want to delve into the differences between action and procedural teachers. So as to put them on an equal footing, we used for all teachers demonstration datasets of limited sizes. The demonstrations in the physical version of the action and procedural teachers reach the same outcomes for $\Omega_1, \Omega_2, \Omega_3$ and $\Omega_4$. An extra action teacher provides demonstrations for the simplest outcome space $\Omega_0$:

- ActionTeacher0 ($\Omega_0$): 9 demos of action primitives;
- ActionTeacher1 ($\Omega_1$): 7 demos of size 2 actions;
- ActionTeacher2 ($\Omega_2$): 7 demos of size 2 actions;
- ActionTeacher3 ($\Omega_3$): 32 demos of size 4 actions;
- ActionTeacher4 ($\Omega_4$): 7 demos of size 4 actions;

The demonstrations for the procedural teachers correspond to the subgoals reached by the action primitives of the action teachers. The procedural teachers have the same number of demonstrations as their respective action teachers.

The action teachers were provided to the SGIM-ACTS learner, while the SGIM-PB algorithm had all procedural teachers and ActionTeacher0. No teacher was provided for the most complex outcome space $\Omega_5$, as to compare the autonomous exploration capability of both learners.

In both setups, the number of demonstrations was chosen arbitrarily small, and the higher the dimensionality of the outcome space it teaches, the more demonstrations it can offer. The demonstrations were chosen so as to cover uniformly the reachable outcome spaces.

### *4.4. Evaluation Method*

To evaluate our algorithm, we created a testbench set of goals uniformly covering the outcome space (the evaluation outcomes are different from the demonstration outcomes). It has 29,200 goals for the real robot and 19,200 goals for simulated version. The evaluation consists in computing mean Euclidean distance between each of the testbench goals and their nearest neighbour in the learner memory ($d_{thres}$= 5).The evaluation is repeated regularly across the learning process.

Then to assess the efficiency of our algorithm in the simulation setup, we are comparing the averaged results of 10 runs of 25,000 learning iterations (each run took about a week to proceed) of the following algorithms :

- RandomAction: random exploration of the action space $\Pi^{\mathbb{N}}$;
- IM-PB: autonomous exploration of the action $\Pi^{\mathbb{N}}$ and procedural space $\Omega^2$ driven by intrinsic motivation;
- SGIM-ACTS: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration of the action space $\Pi^{\mathbb{N}}$ and mimicry of any action teacher;
- SGIM-PB: interactive learner driven by intrinsic motivation. Has autonomous exploration strategies (of the action $\Pi^{\mathbb{N}}$ or procedural $\Omega^2$ space) and mimicry ones for any procedural teacher and ActionTeacher0;
- Teachers: non-incremental learner only provided with the combined knowledge of all the action teachers.

For the physical setup, we are only comparing 1 run of SGIM-ACTS and SGIM-PB (20,000 iterations or one month of trials for each), as to delve deeper into the study of the interactive strategies and their impact.

The codes used are available at https://bitbucket.org/smartan117/sgim-yumi-simu (simulated version), and at https://bitbucket.org/smartan117/sgim-yumi-real (physical one).

## 5. Results

### *5.1. Task Decomposition and Complexity of Actions*

#### 5.1.1. Task Hierarchy Discovered

To check whether SGIM-PB is capable of learning the task hierarchy used to build the setup (see Figure 2), after an exploration of 25,000 compound actions, we tested how it reaches each of the goals of the testbench. More precisely, to show the task hierarchy learned by the algorithm, we reported the types of procedures SGIM-PB uses to reach goals of each goal types and plot in Figure 5 the histogram of the procedures used for $\Omega_1$ and $\Omega_4$ for the simulated (left column) and for $\Omega_1$, $\Omega_4$ and $\Omega_5$ for physical setup (right column). We can see that in both setups, the most associated procedural space with each outcome space corresponds to the designed task hierarchy, namely, $\Omega_1$ uses procedures mostly procedures $(\Omega_0, \Omega_0)$ and $\Omega_4$ uses $(\Omega_1, \Omega_2)$ and $\Omega_5$ uses $(\Omega_4, \Omega_0)$. It also corresponds to the hierarchy demonstrated by the procedural teachers. It was even capable to learn the task hierarchy in the absence of provided teacher for the outcome space $\Omega_5$ in the physical setup. **The SGIM-PB learner was able to learn the task hierarchy of the setup using the procedure framework.** The procedure repartition is not perfect though, as other subgoals are also used. For instance, procedures $(\Omega_1, \Omega_2)$ were associated to the $\Omega_1$ goals. Although they correctly reach the goal tasks, they introduce longer sequences of actions and are thus suboptimal.

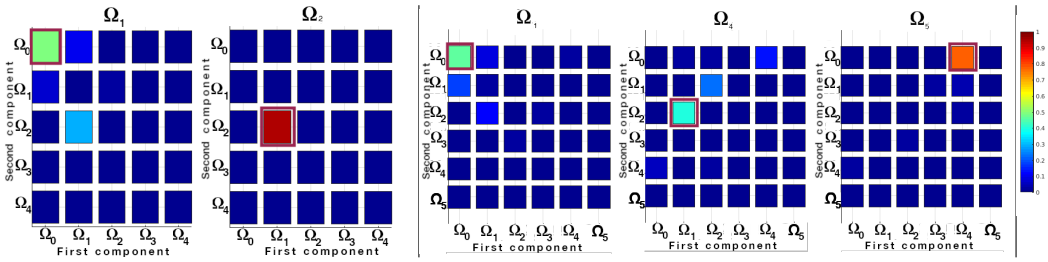



**Figure 5.** Task hierarchy discovered by the SGIM-PB learners: this represents for 3 complex outcome space the percentage of time each procedural space would be chosen for the physical setup (left) and the simulated setup (center). The percentage scale is represented in the colorbar (right).

#### 5.1.2. Length of the Chosen Actions

Indeed, this internal understanding of task hierarchy translates into visible actions, in particular the length of action primitive sequences which should adapt to the level of hierarchy of the goal task. We analysed which action size is chosen by the local action optimization function of SGIM-PB, for each goal task of the evaluation testbench. The results are shown on Figure 6 for each outcome space. In both setups, **SGIM-PB was able to scale the complexity of its actions**: using mostly primitive and sequences of 2 primitives for $\Omega_0$; sequences of 2 primitives for $\Omega_1$ and $\Omega_2$; of 4 primitives for $\Omega_4$ and $\Omega_3$; and of 5 primitives for $\Omega_5$ on the physical setup. It is not perfect as SGIM-PB partially associated the $\Omega_0$ outcome space with 2-action primitives when single action primitives were sufficient. Because all tasks require the robot to touch the table, and thus has an outcome $\Omega_0$ on the environment, all the complex actions used for the other tasks could be associated to reach $\Omega_0$: the redundancy of this model makes it harder to select the optimal action size.

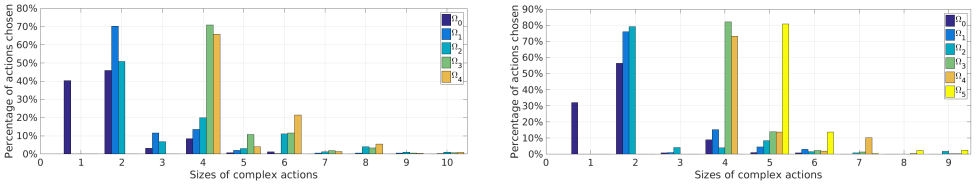



**Figure 6.** Histogram of the action size of actions chosen by SGIM-PB for each outcome space. Test results on the testbench tasks on the simulation (**left**) and physical setups (**right**).

This understanding of task complexity and task hierarchy also leads to a better performance of SGIM-PB. Figure 7 shows the global evaluation results of all tested algorithms for the simulated version. It plots the test mean error made by each algorithm to reach the goals of the benchmark with respect to the number of actions explored during learning. SGIM-PB has the lowest level of error compared to the other algorithms. Thus **SGIM-PB learns with better precision**. This is due to its transfer of knowledge from simple tasks to complex tasks, after learning task hierarchy, owing to both its use of imitation learning and the procedure representation.

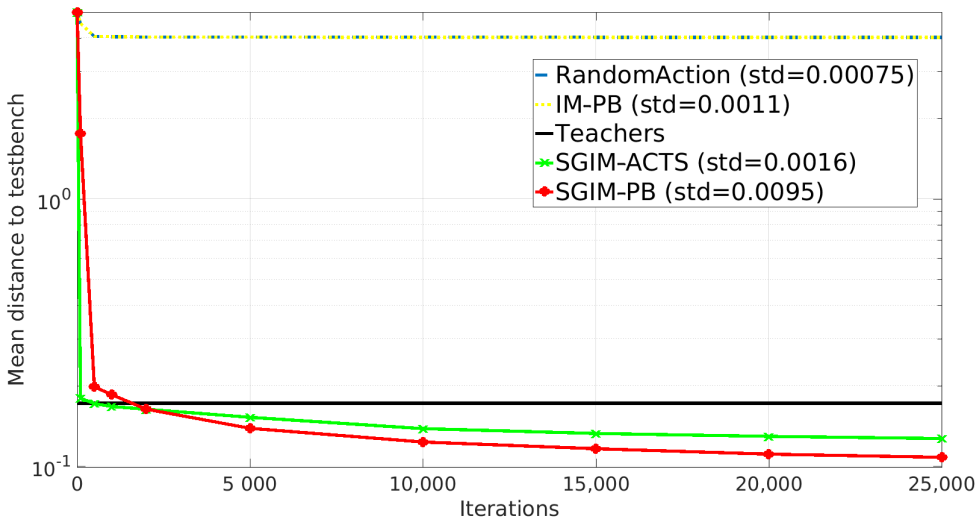



**Figure 7.** Evaluation of all algorithms throughout the learning process for the simulation setup, final standard deviations are given in the legend.

### *5.2. Imitation Learning Bootstraps the Learning of Procedures: Role of Imitation*

First, let us examine the role of imitation by contrasting the algorithms with active imitation (SGIM-PB and SGIM-ACTS) with the autonomous learners without imitation (RandomAction and IM-PB). IM-PB is the variation of SGIM-PB with only autonomous learning, without imitation of actions or imitation of procedures. In Figure 7, both autonomous learners have higher final levels of error than the active imitation learners, which shows the advantage of using social guidance. Besides, both SGIM-ACTS and SGIM-PB have error levels dropping below that of the teachers, showing they learned further than the provided action demonstrations: the combination of autonomous exploration and imitation learning improves the learner's performance beyond the performance of the teachers.

Figure 8 plots the evaluation on the simulated setup for each type of tasks. While all algorithms have about the same performance on the simple task $\Omega_0$, we notice a significant difference for the complex tasks in $\Omega_1, \Omega_2, \Omega_3$ or $\Omega_4$ between the autonomous and the active imitation learners. The autonomous learners were not able to even reach a goal in the complex subspaces. In particular, the difference between IM-PB and SGIM-PB means that imitation is necessary to learn complex tasks, it is not only a speeding-up effect.

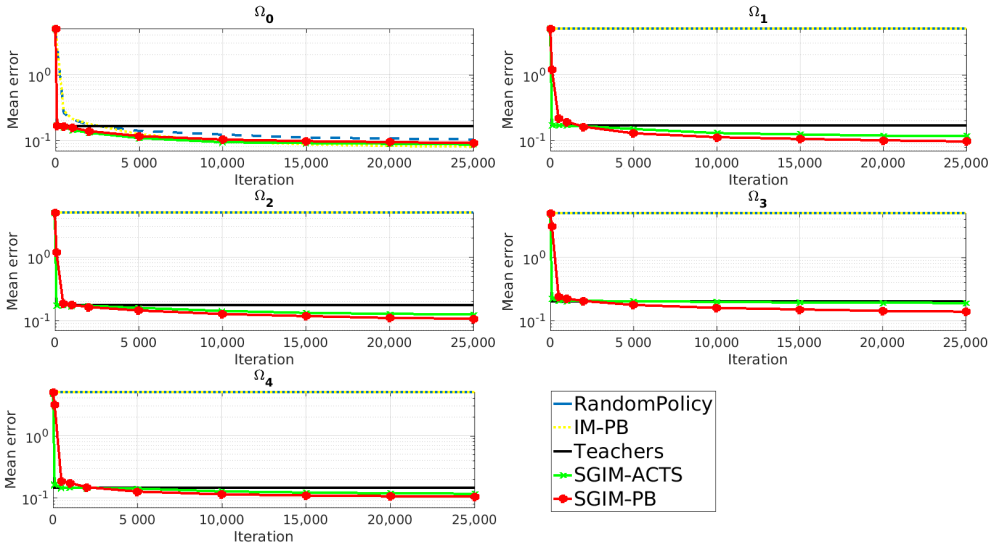



**Figure 8.** Evaluation of all algorithms per outcome space on the simulation setup (RandomAction and IM-PB are superposed on all except for $\Omega_0$).

These results show that the imitation strategies have improved the performance from the beginning of the learning process, and this improvement is more visible for complex tasks. **More than a speeding effect, imitation enables the robot to reach the first goals in the complex task subspaces, to be optimised and generalised later on by autonomous exploration, so that SGIM-PB is not bound by the limitations of the demonstration dataset**.

### *5.3. Procedures Are Essential for Higher Level of Hierarchy Tasks: Role of Procedures*

Second, let us examine the role of procedures, both for imitation and for autonomous exploration.

#### 5.3.1. Demonstrations of Procedures

To analyse the difference between the different imitation strategies, i.e between imitation of action primitives and imitation of procedures, we can compare the algorithms SGIM-PB and SGIM-ACTS. While SGIM-PB has procedures teachers for the complex tasks and can explore the procedure space to learn task decomposition, SGIM-ACTS has action teachers and does not have the procedure representation to learn task decomposition. Instead SGIM-ACTS explores the action space by choosing a length for its action primitive sequence, then the parameters of the primitives.

In Figure 7, we see that SGIM-PB is able to outperform SGIM-ACTS after only 2000 iterations, which suggests that procedural teachers can effectively replace action teachers for complex tasks. More precisely, as shown in Figure 8, for tasks $\Omega_0$ where SGIM-ACTS and SGIM-PB have the same action primitive teacher ActionTeacher0, there is no difference in performance, and SGIM-PB was outperforming SGIM-ACTS on all the complex tasks, particularly $\Omega_3$.

To understand the learning process of SGIM-PB that leads to this difference in performance, let us look at the evolution of the choices of each outcome space (Figure 9) and strategy (Figure 10). The improvement of performance of SGIM-PB compared to the other algorithms can be explained in Figure 9 by its choice of task $\Omega_3$ in the curriculum for iterations above 10,000 after an initial phase where it explores all outcome spaces. Figure 10, we notice that SGIM-PB chooses mainly as strategies: ProceduralTeacher3 among all imitation strategies, and Autonomous procedures among the autonomous exploration strategies. The histogram of each task-strategy combination chosen for the whole learning process in Figure 11 confirms that ProceduralTeacher3 was chosen the most frequently among imitation strategies specifically for tasks $\Omega_3$, but SGIM-PB used most extensively Autonomous procedures. Thus **SGIM-**

**PB performance improvement is correlated with its procedure space exploration with both Autonomous procedures and procedural imitation strategies**.

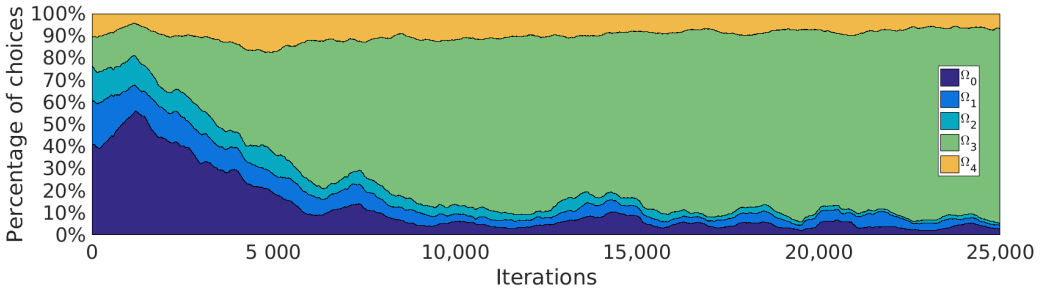



**Figure 9.** Evolution of choices of tasks for the SGIM-PB learner during the learning process on the simulation setup.

This comparison between SGIM-PB and SGIM-ACTS on simulation is confirmed on the physical setup. The global evaluation in Figure 12 shows a significant gap between the two performances. SGIM-PB even outperforms SGIM-ACTS after only 1000 iterations, which suggests that procedural teachers can be more effective than action teachers for complex tasks.

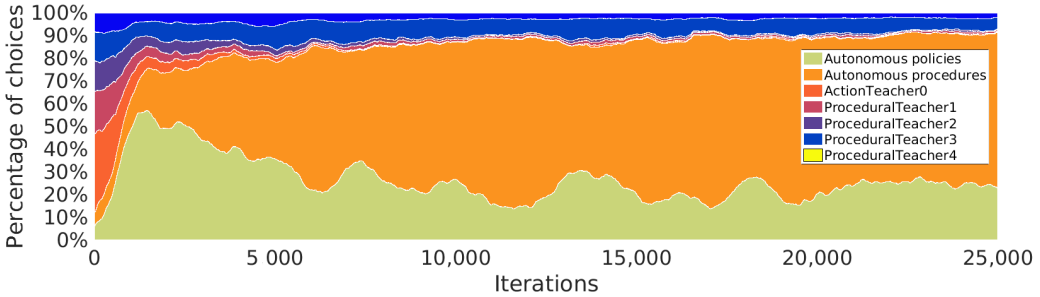



**Figure 10.** Evolution of choices of strategies for the SGIM-PB learner during the learning process on the simulation setup.

The performance per type of tasks in Figure 13 shows that like in the simulation setup, there is little difference for the simple tasks $\Omega_0$ and $\Omega_1$, and there is more difference on the complex tasks $\Omega_3$ and $\Omega_4$. The more complex the task, the more SGIM-PB outperforms SGIM-ACTS. These confirm that **procedural teachers are better adapted to tackle the most complex and hierarchical outcome spaces**.

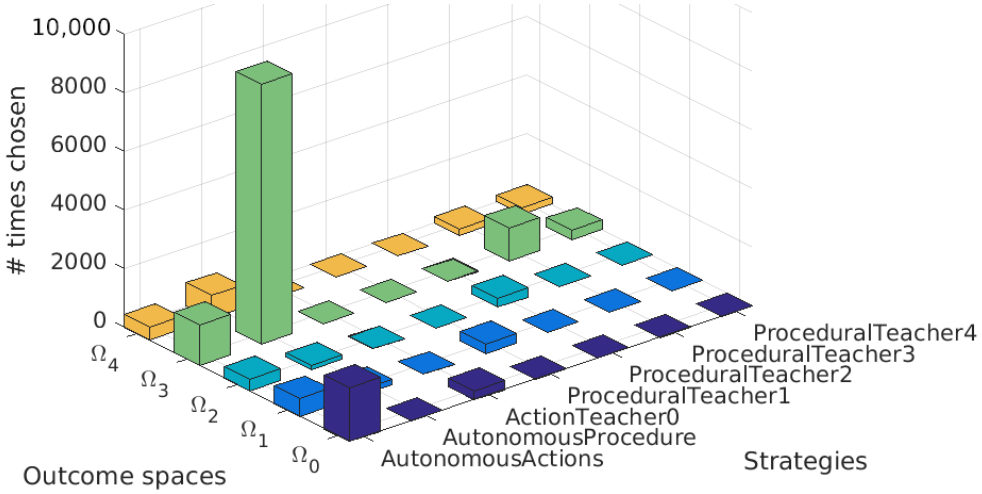



**Figure 11.** Choices of strategy and goal outcome for the SGIM-PB learner during the learning process on the simulation setup.

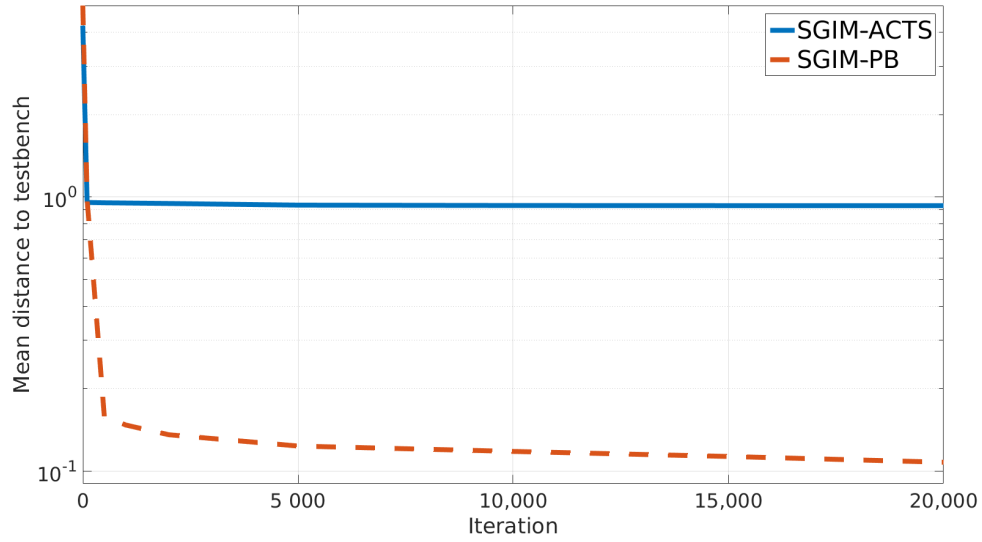



**Figure 12.** Evaluation of all algorithms throughout the learning process for the physical setup.

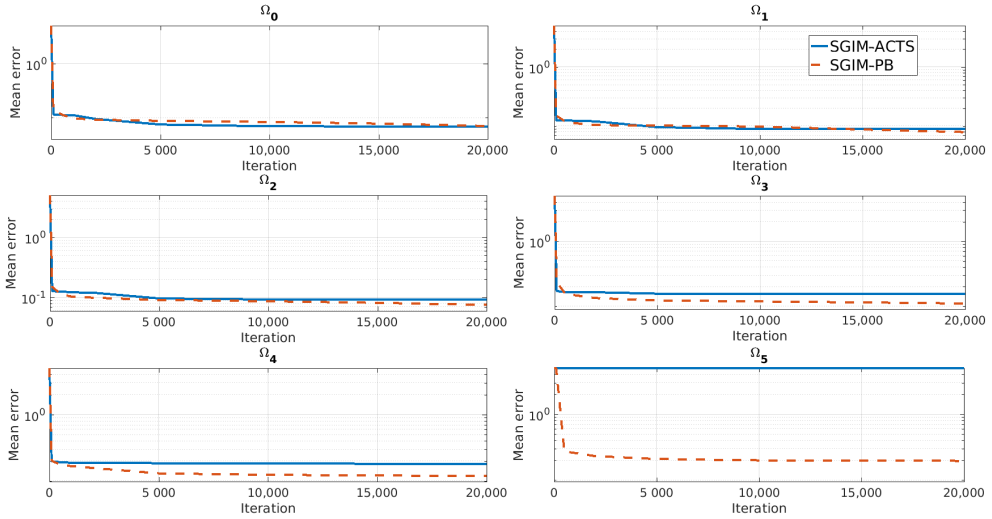



**Figure 13.** Evaluation for each outcome space of the physical Yumi setup.

5.3.2. Autonomous Exploration of Procedures

Moreover, in the real yumi setup, we have a supplementary type of tasks $\Omega_5$, which are also the highest level of hierarchy tasks. For $\Omega_5$, no action or procedural teacher was provided to SGIM-PB or SGIM-ACTS, therefore we can contrast the specific effects of autonomous exploration of procedures to autonomous exploration of the action space. Figure 13 shows that the performance of SGIM-ACTS is constant for $\Omega_5$, it is not able to reach any task in $\Omega_5$ even once, while SGIM-PB, owing to the capability of the procedure framework to reuse the knowledge acquired for the other tasks, is able to explore in this outcome space.

To understand the reasons for this difference, let us examine the histogram of the strategies chosen per task in Figure 14. For $\Omega_5$, SGIM-PB uses massively procedure space exploration compared to action space exploration. Owing to autonomous procedure exploration, SGIM-PB can thus learn complex tasks by using the decomposition of tasks into known subtasks.

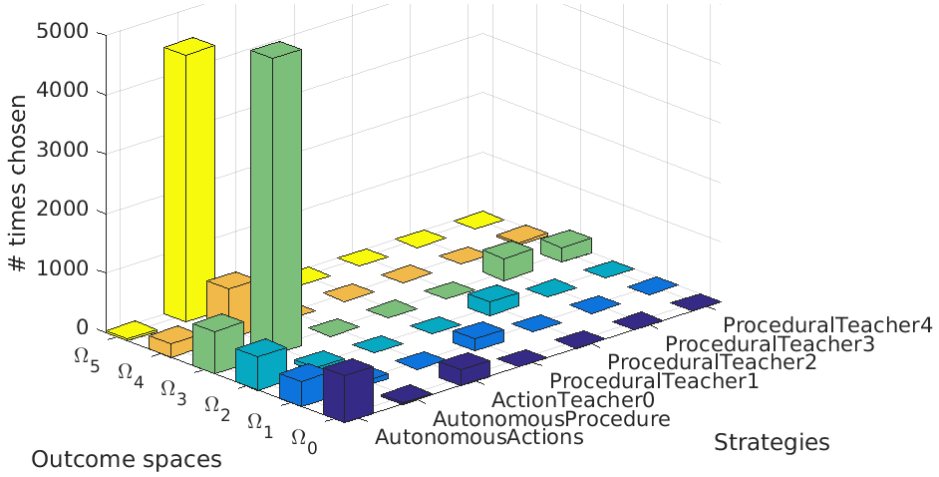



**Figure 14.** Choices of strategy and goal outcome for the SGIM-PB learner during the learning process on the physical setup.

This highlights the **essential role of the procedure representation and the procedure space exploration by imitation but also by autonomous exploration, in order to learn high-level hierarchy tasks, which have sparse rewards**.

*5.4. Curriculum Learning by SGIM-PB*

Given the properties of imitation and procedures that we outlined, is SGIM-PB capable of choosing the right strategy for each task to build a curriculum starting from simple tasks before complex tasks?

Figure 14 shows that SGIM-PB uses more procedural exploration and imitation than action exploration or imitation for $\Omega_3$ and $\Omega_4$, compared to the more simple tasks. A coupling appears between simple tasks and action space exploration on the one hand, and complex tasks and procedure exploration on the other hand. Moreover, for imitation, **SGIM-PB was overall able to correctly identify the most adapted teacher to each outcome space**. Their only suboptimal choice is to use the procedural teacher built for $\Omega_4$ to explore $\Omega_3$. This mistake can be explained as both outcome spaces have the same task decomposition (see Figure 2).

Likewise, for the simulation setup, in Figure 11, the histogram of explored task-strategy combinations confirms that the learner uses mostly autonomous exploration strategies. It uses mostly action exploration for the simplest outcome spaces ($\Omega_0, \Omega_1, \Omega_2$), and procedure exploration for the most complex outcomes ($\Omega_3, \Omega_4$). This shows the complementarity of action and procedures for exploration in an environment with a hierarchical set of outcome spaces. We can see for each task subspace, the proportion of

imitation used. While for $\Omega_0$, SGIM-PB uses the strategy AutonomousActions five times more than ActionTeacher0, the proportion of imitation increases for the complex tasks. Imitation seems to be required more for complex tasks. For the simplest outcome spaces $(\Omega_0, \Omega_1, \Omega_2)$, it uses mostly action exploration and procedure exploration for the most complex ones $(\Omega_3, \Omega_4)$. From Figure 11 , we can also confirm that for each task, the teacher most requested is specialised in the goal task, the only exceptions are ProceduralTeacher3 and ProceduralTeacher4 who are specialised in different complex tasks but that use the same subtask decomposition, thus demonstrations of ProceduralTeacher3 has effects on $\Omega_4$ and vice versa. The choices shown in the histogram show that SGIM-PB has spotted the teacher's domain of expertise.

Let us analyse the evolution of the time each outcome space (Figure 9) and strategy (Figure 10) is chosen during the learning process of SGIM-PB on the simulated setup. In Figure 9, its self-assigned curriculum starts by exploring the most simple task $\Omega_0$ until 1000 iterations. In Figure 10, we see that this period corresponds mainly to the use of the strategy Autonomous policies, relying on itself to acquire its body schema. Then it gradually switches to working on the most complex outcome space $\Omega_3$ (the highest dimension) and marginally more on $\Omega_4$ while preferring autonomous procedures and marginally the teachers for $\Omega_3$ and $\Omega_4$. In contrast, the strategy ActionTeacher0 decreases, SGIM-PB does not use action imitation any more. **SGIM-PB switches from imitation of action primitives to procedures, and most of all it turns to the strategies autonomous policies and autonomous procedures**.

For the physical setup, the evolution of outcome spaces (Figure 15) and strategies (Figure 16) chosen are more difficult to analyse. However, they show the same trend from iterations 0 to 10,000: the easy outcome spaces $\Omega_0, \Omega_1, \Omega_2$ are more explored in the beginning before being neglected after 1000 iterations to explore the most complex outcome spaces $\Omega_3, \Omega_5$. Imitation is mostly used in the beginning of the learning process, whereas later in the curriculum, autonomous exploration is preferred. The autonomous exploration of actions strategy was also less used. However, the curriculum after 10,000 is harder to analyse : at two instances (around 13,000 and a second time around 17,000 iterations), SGIM-PB switched to autonomous exploration of actions while exploring simpler outcome spaces $\Omega_0, \Omega_1, \Omega_2$. This might mean the learner needs to consolidate its basic knowledge on the basic tasks before being able to make further progress in the complex tasks.

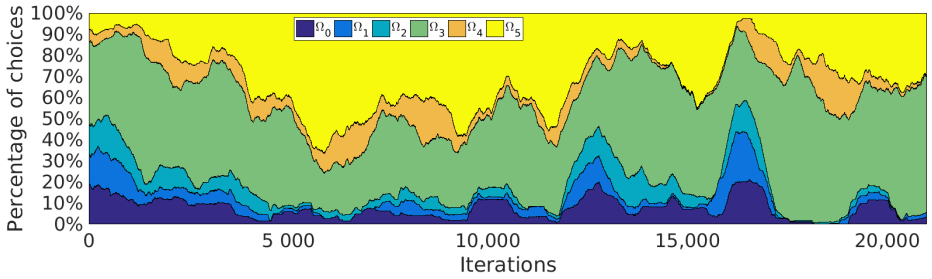



**Figure 15.** Evolution of choices of tasks for the SGIM-PB learner during the learning process on the physical setup.

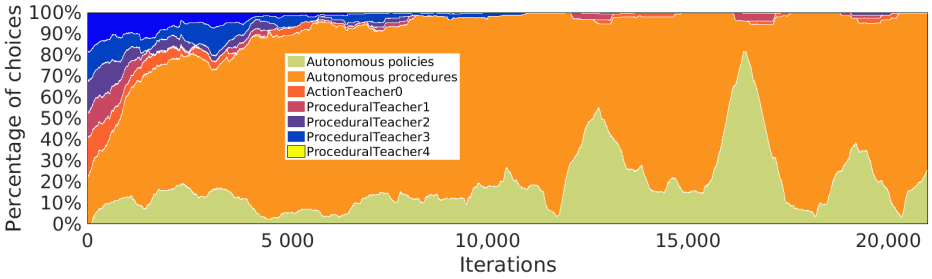



**Figure 16.** Evolution of choices of strategies for the SGIM-PB learner during the learning process on the physical setup.

### *5.5. Data Efficiency of Active Learning: Active Imitation Learning vs Batch Transfer*

In this section, we explore the possibility to transfer a set of procedures as a batch at the beginning of the learning process, as opposed to active requests from the learner throughout the learning process. We consider a learner and a teacher with different embodiments working on the same tasks. While transfer of knowledge of actions can not be reused straightforward, how can the knowledge of procedures be exploited?

In our example, we consider new learners trying to explore the interactive table using the left arm of Yumi, while benefitting from the knowledge acquired from a right arm Yumi. We call this simulation setup the *left-arm setup*. We extracted a dataset $D_3$ composed of the procedures and their corresponding reached outcomes $((\omega_i, \omega_j), \omega_r))$ taken from the experience memory of a SGIM-PB learner that has trained on the right arm for 25,000 iterations (taken from the runs of SGIM-PB on the simulated setup). To analyse the benefits of batch transfer before learning, we run in the simulated setup (see Section 4.1), two variants of SGIM-PB 10 times, with for each 10,000 iterations:

- Left-Yumi-SGIM-PB: the classical SGIM-PB learner using its left arm, using from the exact same strategies as on the simulated setup, without any procedure transferred;
- Left-Yumi-SGIM-TL: a modified SGIM-PB learner, benefiting from the strategies used on the simulated setup, and which benefits from the dataset $D_3$ as a Transferred Lump at the initialisation phase: $Memo \leftarrow \{((\omega_i, \omega_j), \omega_r)\}$ at the beginning their learning process. No actions are transferred, and the transferred data are only used for computing local exploration of the procedural space, so they don't impact the interest model nor the test evaluations reported in the next section.

All procedural teachers propose the same procedures as in Section 4.1 for the right-handed SGIM-PB learner. The ActionTeacher0, proposing action demonstrations for reaching $\Omega_0$ outcomes was changed to left-handed demonstrations.

#### 5.5.1. Evaluation Performance

Figure 17 shows the global evaluation of Left-Yumi-SGIM-PB and Left-Yumi-SGIM-TL. We can see that, even though the learning process seems quicker before 500 iterations for the Left-Yumi-SGIM-TL learner, both learners quickly learn at the same rate as shown by their overlapping evaluation graphs. If we look at Figure 18 which shows this evaluation broken down for each task, we see the same phenomenon for $\Omega_2, \Omega_3$ and $\Omega_4$. This seems to indicate, that active imitation using the small demonstration datasets are enough for Left-Yumi-SGIM-PB to tackle this setup, while the huge dataset of transferred procedures don't give Left-Yumi-SGIM-TL an edge other than a slightly better performance for the complex tasks at the initial phase.

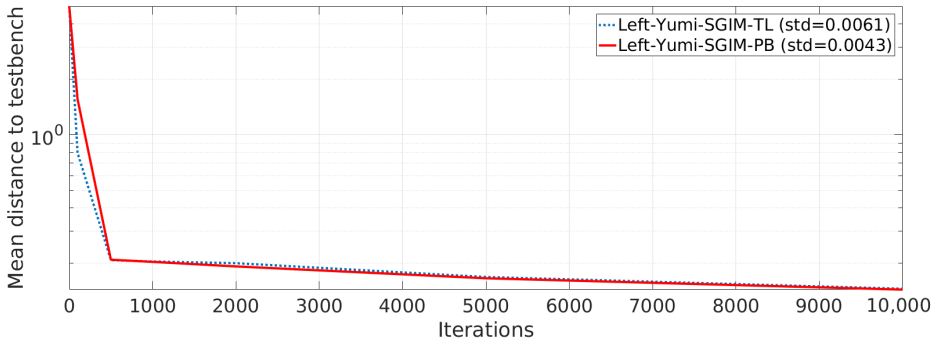



**Figure 17.** Evaluation of all algorithms throughout learning process for the transfer learning setup, final standard deviations are given in the legend.

#### 5.5.2. Procedures Learned

To analyse what task hierarchy both learners have discovered at the end of their learning process, we plotted Figure 19. We can see that no learner is clearly better at

discovering the setup task hierarchy (see Figure 2): Left-Yumi-SGIM-PB decomposes more often tasks $\Omega_1$ as $(\Omega_0, \Omega_0)$, but slightly less often $\Omega_3$ or $\Omega_4$ as $(\Omega_1, \Omega_2)$. For $\Omega_3$, Left-Yumi-SGIM-PB also uses another procedure: $(\Omega_2, \Omega_1)$ which is also valid. To position both objects, the robot can start by either object 1 or object 2. However, if we take into account the task hierarchy learned by the transfer dataset which is fed to Left-Yumi-SGIM-TL before its learning process, we can see that Left-Yumi-SGIM-TL has learned almost exactly the same task hierarchy than the transfer dataset, which indicates its influence owing to the large number of procedures transferred: procedures from 25,000 iterations were transferred compared to the new procedures explored by Left-Yumi-SGIM-TL during only 10,000 iterations. Hence Left-Yumi-SGIM-TL has the same defects and qualities than the transfer dataset in terms of the task hierarchy discovered. For instance, Left-Yumi-SGIM-TL uses the inefficient procedure $(\Omega_1, \Omega_2)$ to reach $\Omega_1$ more often than Left-Yumi-SGIM-PB. It wasn't able to overstep the defects of the transfer dataset.

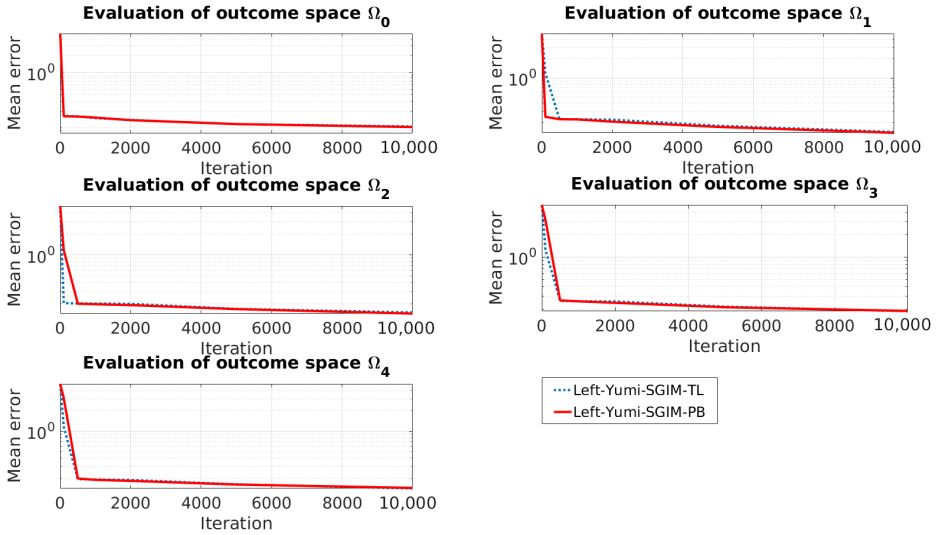



**Figure 18.** Task evaluation of all algorithms throughout learning process for transfer learning setup.

### 5.5.3. Procedures Used during Learning

If we look at the procedures that were actually tried out by the learner during its learning process for reaching each goal outcome (see Figure 20), we can see first that for all versions of the algorithm and all types of tasks except $\Omega_1$, the procedures most used during the learning phase correspond to the ground truth. Thus intrinsic motivation has oriented the exploration towards the relevant task decompositions. Besides, for all types of task, Left-Yumi-SGIM-TL tends to explore more the procedures in $(\Omega_1, \Omega_2)$ than Left-Yumi-SGIM-PB. This difference can also be explained by the predominance of this procedure in the transfer dataset $D_3$. Both of them also tried a lot of procedures from the $\Omega_0^2$ procedural space. It confirms that Left-Yumi-SGIM-TL was not able to filter out the defects of its transfer dataset in terms of the task hierarchy provided.

### 5.5.4. Strategical Choices

Analysing the learning process (Figures 21–24) of both learners, we can see that they are very similar. Both learners start by performing a lot of imitation of the available teachers coupled with an exploration of all outcome types, until 1500 and 2000 iterations for respectively Left-Yumi-SGIM-TL and Left-Yumi-SGIM-PB. This difference in timing can be caused by the transferred dataset $D_3$, but is not very significant. Then they focus more on the autonomous exploration of the action space to reach the $\Omega_0$ outcome subspace before gradually working on more complex outcome spaces while performing more an more autonomous exploration of the procedural space. However, the Left-Yumi-SGIM-TL learner seems to abandon its initial imitation phase faster than Left-Yumi-SGIM-PB (about 1000 iterations faster), and also quickly starts working on the more complex $\Omega_3$

outcome space with the strategy autonomous exploration on procedures. This initial faster maturation seems perhaps too fast as Left-Yumi-SGIM-TL reverts to working on $\Omega_0$ with autonomous actions afterwards : we see two other peaks of this choice of combination at 5500 iterations and 9500 iterations. On the contrary, Left-Yumi-SGIM-PB seems to converge more steadily towards its final learning phase.

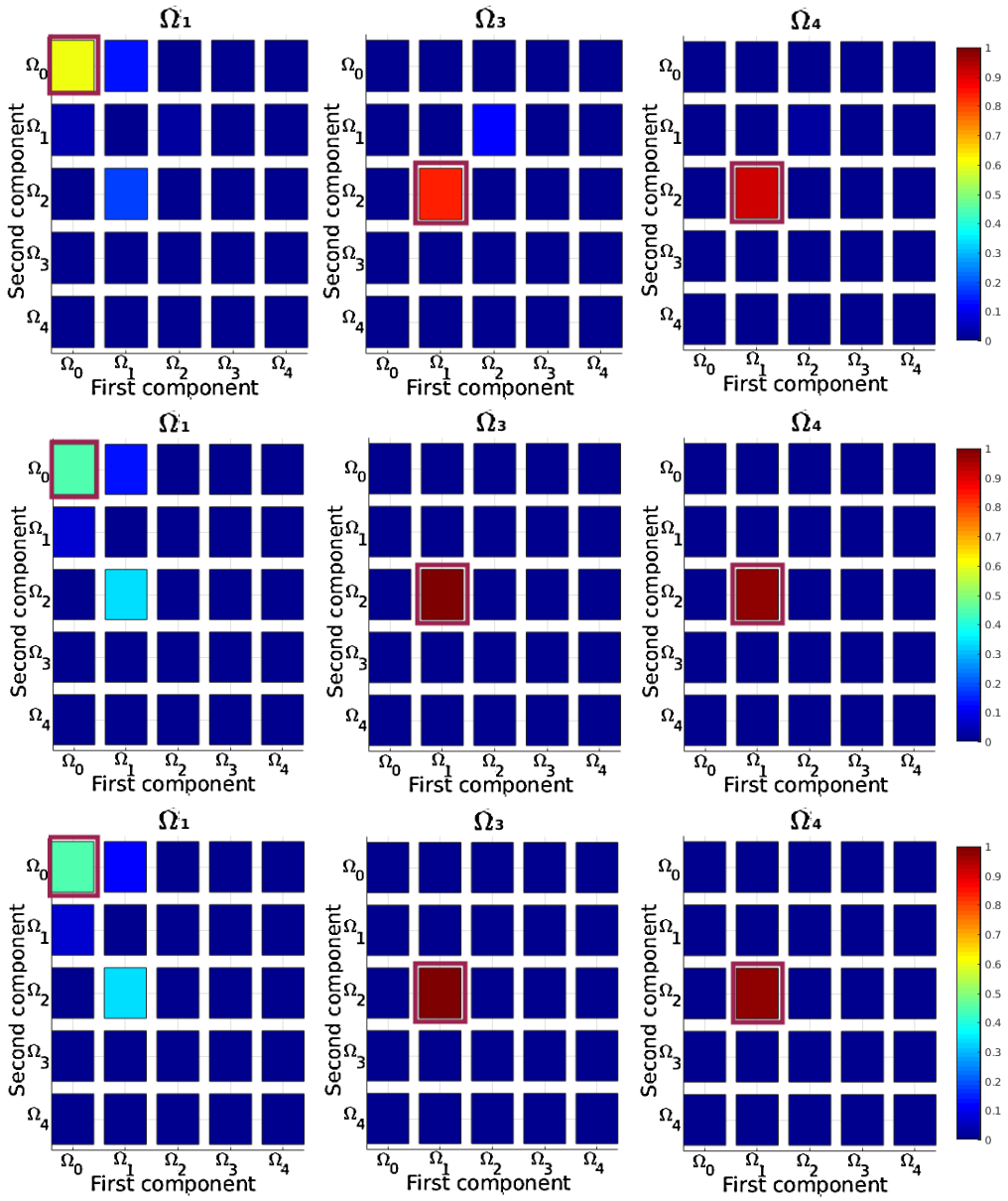



**Figure 19.** Task hierarchy discovered by the learners on left-arm setup: this represents for outcome spaces $\Omega_1, \Omega_3$ and $\Omega_4$ the percentage of time each procedural space is chosen during test to reach goals of the testbench by Left-Yumi-SGIM-PB (**1st row**), the transfer dataset (**2nd row**) and the Left-Yumi-SGIM-TL learner (**3rd row**). The ground truth hierarchy is indicated by a red square.

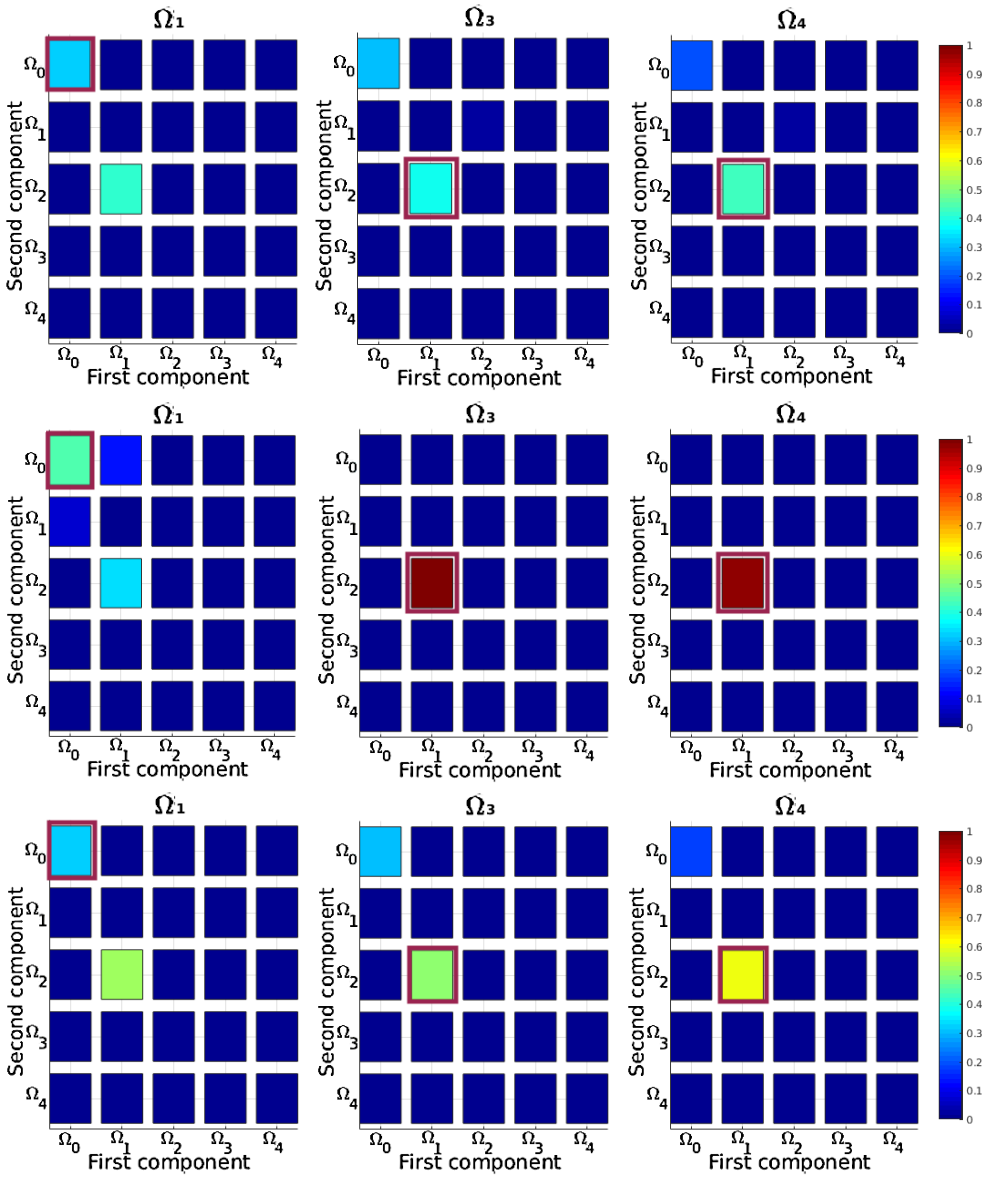



**Figure 20.** Procedures used during the exploration phase on left-arm setup: this shows for outcome spaces $\Omega_1, \Omega_3$ and $\Omega_4$ the percentage of times each procedural space was chosen during learning by Left-Yumi-SGIM-PB (**1st row**), and the Left-Yumi-SGIM-TL learner (**3rd row**), procedures discovered by the transfer dataset $\mathcal{D}_3$ are also reminded (**2nd row**). The ground truth hierarchy is indicated by a red square.

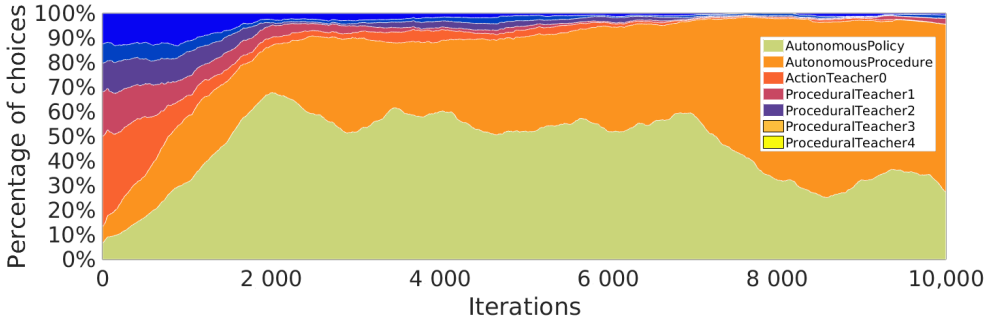



**Figure 21.** Evolution of choices of strategies for the Left-Yumi-SGIM-PB learner during the learning process on left-arm setup.

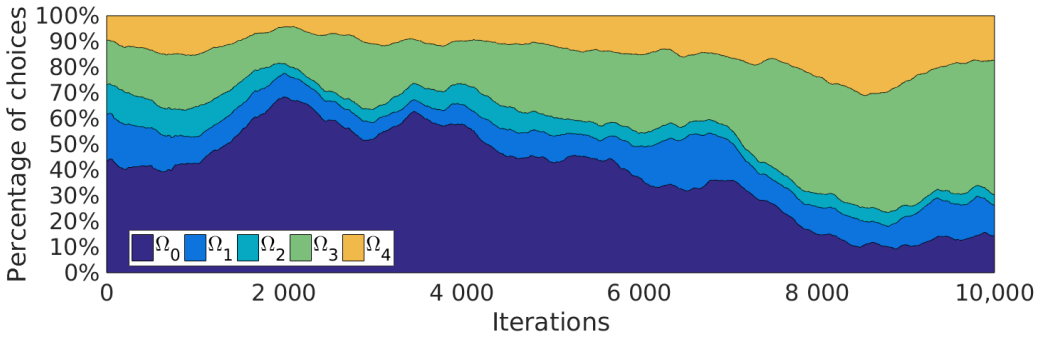



**Figure 22.** Evolution of choices of tasks for the Left-Yumi-SGIM-PB learner during the learning process on left-arm setup.

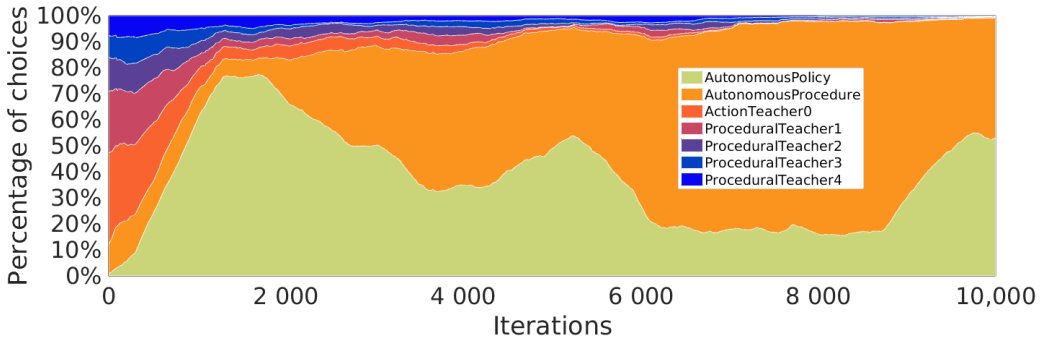



**Figure 23.** Evolution of choices of strategies for the Left-Yumi-SGIM-TL learner during the learning process on left-arm setup.

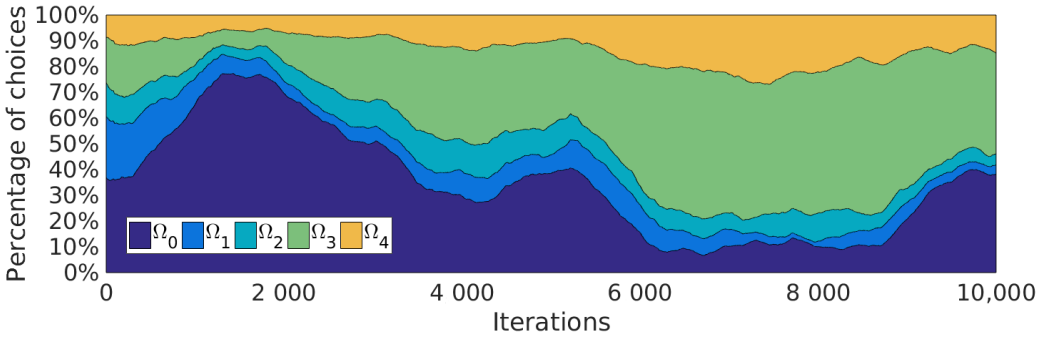



**Figure 24.** Evolution of choices of tasks for the Left-Yumi-SGIM-TL learner during the learning process on left-arm setup.

If we look at the choices of strategy and goal outcomes for the whole learning process (see Figures 25 and 26), we can see that this difference in the learning processes is visible in the number of times each of the two main combinations of task and goal outcome space was chosen: Left-Yumi-SGIM-TL favors more working autonomously on procedures for exploring $\Omega_3$ whereas Left-Yumi-SGIM-PB worked more on autonomous exploration of actions for $\Omega_0$.

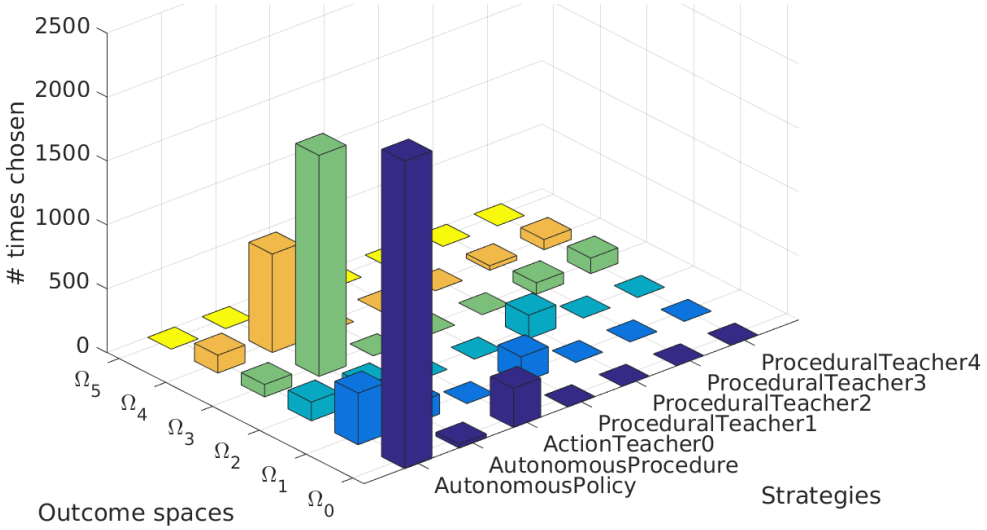



**Figure 25.** Choices of strategy and goal outcome for the Left-Yumi-SGIM-PB learner on left-arm setup.

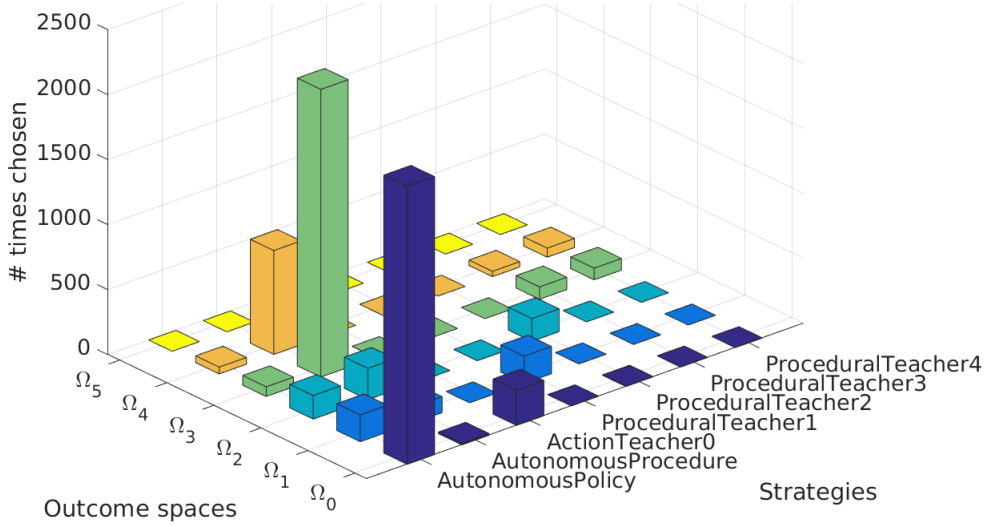



**Figure 26.** Choices of strategy and goal outcome for the Left-Yumi-SGIM-TL learner on the left-arm setup.

These results seem to indicate that if a transfer of knowledge about difficult tasks takes place before easy tasks are learned, it can disturb the learning curriculum by changing the learner's focus on difficult tasks. The learner needs to realise that the gap in knowledge is too high, and give up these difficult tasks, to re-focus on learning the easy subtasks. Thus, demonstrations given throughout the learning process and adapted to the development of the learner seem more beneficial than a batch of data given at a single point of time, despite a larger amount of data. These results show that **procedure demonstrations can be effective for robots of different embodiments to learn complex tasks; and active requests of few procedure demonstrations are effective to learn task hierarchy**. They were not significantly improved by more data added at initialisation in terms of error, exploration strategy or autonomous curriculum learning.

## 6. Discussion

The experimental results highlight the following properties of SGIM-PB:

- procedures representation and task composition become necessary to reach tasks of higher hierarchy. It is not a simple bootstrapping effect.
- transfer of knowledge of procedures is efficient for cross-learner transfer of knowledge even when they have different embodiments.

- active imitation learning of procedures is more advantageous than imitation from a dataset provided from the initialization phase.

The performance of SGIM-PB stems from its tackling several aspects of transfer of knowledge, and relies on our proposed representation of compound actions that allows hierarchical reinforcement learning.

### *6.1. A Goal-Oriented Representation of Actions*

Our representation of actions allows our learning algorithm to **adapt the complexity of its actions to the complexity of the task at hand**, whereas other approaches using via points [49], or parametrised skills [50], had to bound the complexity of their actions. Other approaches like [51] also use a temporally abstract representation of actions: options. However options are often used in discrete states settings to reach specific states such as bottlenecks, and are generally learned beforehand. On the contrary, our dual representation of skills as action primitive sequences and procedures allow an online learning of an unlimited number of complex behaviours. Our exploitation of the learned action primitive sequence is simplistic and needs improvement though.

Our work proposes a representation of the relationship between tasks and their complexities, by the procedural framework. Comparatively, [25] proposed for tackling a hierarchical multi-task setting, to learn action primitives and use planning to recursively chain skills. However, that approach does not build a representation of a sequence of action primitives, and planning grows slower as the environment is explored. A conjoint use of planning techniques with an unrestrained exploration of the procedural space could be an interesting prospect and extension of our work.

In this article, we tackled the learning of complex control tasks using sequences of actions of unbounded length. Our main contribution is a **dual representation of compound actions in both action and outcome spaces**. We showed its impact on autonomous exploration but also on imitation learning: SGIM-PB learns the most complex tasks by autonomous procedural space exploration, and can benefit more from procedural demonstrations for complex tasks and from motor demonstrations for simple tasks, confirming the results on a more simple setup in [35]. Our work demonstrates the gains that can be achieved by requesting just a small amount of demonstration data with the right type of information with respect to the complexity of the tasks. Our work should be improved by a better exploitation algorithm of the low-level control model and can be speeded up by adding planning.

### *6.2. Transfers of Knowledge*

We have tested our algorithm in the classical setups of transfer of knowledge : **cross-task transfer, cross-learner transfer and by imitation learning**. We have shown that **SGIM-PB can autonomously determine** the main questions of Transfer Learning as theorised in [5]:

- What information to transfer? For compositional tasks, a demonstration of task decomposition is more useful than a demonstration of an action, as it helps bootstrap cross-task transfer of knowledge. Our case study shows a clear advantage of procedure demonstrations and procedure exploration. On the contrary, for simple tasks, action demonstrations and action space exploration show more advantages. Furthermore, for cross-learner transfer, especially when the learners have different embodiments, this case study indicates that demonstrations of procedures are still helpful, whereas demonstrations of actions are no longer relevant.
- How to transfer? We showed that decomposition of a hierarchical task, through procedure exploration and imitation, is more efficient than learning directly action parameters, i.e., interpolation of action parameters. This confirms the results found in a more simple setup in [35].
- When to transfer? Our last setup shows that an active imitation learner asking adapted demonstrations, as its competence increases, performs almost better than when it is

given a significantly larger batch of data at initialisation time. More generally for a less data-hungry transfer learning algorithm, the transferred dataset should be given to the learner in a timely manner so that the information is adapted to the current level of the learner, i.e., its zone of proximal development [52]. This advantage has already been shown by an active and strategic learner—SGIM-IM [53], a simpler versions of SGIM-PB—which had better performance than passive learners for multi-task learning using action primitives.

- Which source of information? Our strategical learner chooses for each task the most efficient strategy between self-exploration and imitation. Most of all, it could understand the domain of expertise of the different teachers and choose the most appropriate expert to imitate. The results of this case study confirms the results found in [35] in a simpler setup and in [38] for learning action primitives.

## 7. Conclusions

We showed that our industrial robot could learn sequences of motor actions of unrestrained size to achieve a field of hierarchically organized outcomes. To learn to control in continuous high-dimensional spaces of outcomes through a continuous infinite dimensionality space of actions, we combined: goal-oriented exploration to enable the learner to organize its learning process in a multi-task learning setting, procedures as a task-oriented representation to build increasingly more complex sequences of actions, active imitation strategies to select the most appropriate information and source of information, and intrinsic motivation as a heuristic to drive the robot's curriculum learning process. All four aspects are combined inside a curriculum learner called SGIM-PB. This algorithm showed the following characteristics through this study:

- Hierarchical RL: it learns online task decomposition on 4 levels of hierarchy using the procedural framework; and it exploits the task decomposition to match the complexity of the sequences of action primitives to the task;
- Curriculum learning: it autonomously switches from simple to complex tasks, and from exploration of actions for simple tasks to exploration of procedures for the most complex tasks;
- Imitation learning: it empirically infers which kind of information is useful for each kind of task and requests just a small amount of demonstrations with the right type of information by choosing between procedural and action teachers;
- Transfer of knowledge: it automatically decides what information, how, when to transfer and which source of information for cross-task and cross-learner transfer learning.

Thus, **our work proposes an active imitation learning algorithm based on intrinsic motivation that uses empirical measures of competence progress to choose at the same time what target task to focus on, which source tasks to reuse and how to transfer knowledge about task decomposition.** Our contributions, grounded in the field of cognitive robotics, are : a new representation of complex actions enabling the exploitation of task decomposition and the proposition for tutors of supplying information on the task hierarchy to learn compound tasks.

This work should be improved by a better exploitation algorithm of the low-level control model and can be speeded up by adding planning methods.

**Author Contributions:** Conceptualization, S.M.N.; Formal analysis, J.Z.; Funding acquisition, D.D.; Investigation, N.D.; Methodology, Jerome Kerdreux; Writing—original draft, N.D.; Writing—review & editing, S.M.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche, European Regional Development Fund (ERDF), Région Bretagne, Conseil Général du Finistère) and by Institut Mines Télécom, received in the framework of the VITAAL project.

**Institutional Review Board Statement:** Not applicable

**Informed Consent Statement:** Not applicable

**Data Availability Statement:** The codes used are available at https://bitbucket.org/smartan117/sgim-yumi-simu (simulated version), and at https://bitbucket.org/smartan117/sgim-yumi-real (physical one).

**Acknowledgments:** This work was supported by Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche, European Regional Development Fund (ERDF), Région Bretagne, Conseil Général du Finistère) and by Institut Mines Télécom, received in the framework of the VITAAL project.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An introduction*; MIT Press: Cambridge, MA, USA, 1998.
2. Zech, P.; Renaudo, E.; Haller, S.; Zhang, X.; Piater, J. Action representations in robotics: A taxonomy and systematic classification. *Int. J. Robot. Res.* **2019**, *38*, 518–562, doi:10.1177/0278364919835020.
3. Elman, J. Learning and development in neural networks: The importance of starting small. *Cognition* **1993**, *48*, 71–99.
4. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*; ACM: New York, NY, USA, 2009; pp. 41–48.
5. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359.
6. Taylor, M.E.; Stone, P. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.* **2009**, *10*, 1633–1685.
7. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9, doi:10.1186/s40537-016-0043-6.
8. Whiten, A. Primate culture and social learning. *Cogn. Sci.* **2000**, *24*, 477–508.
9. Call, J.; Carpenter, M. *Imitation in Animals and Artifacts*; Chapter Three Sources of Information in Social Learning; MIT Press: Cambridge, MA, USA, 2002; pp. 211–228.
10. Tomasello, M.; Carpenter, M. Shared intentionality. *Dev. Sci.* **2007**, *10*, 121–125.
11. Piaget, J. *The origins of intelligence in children (M. Cook, Trans.)*; WW Norton & Co: New York, NY, USA, 1952.
12. Deci, E.; Ryan, R.M. *Intrinsic Motivation and Self-Determination in Human Behavior*; Plenum Press: New York, NY, USA, 1985.
13. Oudeyer, P.Y.; Kaplan, F.; Hafner, V. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Trans. Evol. Comput.* **2007**, *11*, 265–286.
14. Schmidhuber, J. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Trans. Auton. Ment. Dev.* **2010**, *2*, 230–247.
15. Baranes, A.; Oudeyer, P.Y. Intrinsically motivated goal exploration for active motor learning in robots: A case study. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 1766–1773.
16. Rolf, M.; Steil, J.; Gienger, M. Goal Babbling permits Direct Learning of Inverse Kinematics. *IEEE Trans. Auton. Ment. Dev.* **2010**, *2*, 216–229.
17. Forestier, S.; Mollard, Y.; Oudeyer, P. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. *arXiv* **2017**, arXiv:1708.02190.
18. Colas, C.; Fournier, P.; Chetouani, M.; Sigaud, O.; Oudeyer, P.Y. CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In *International Conference on Machine Learning*; Chaudhuri, K., Salakhutdinov, R., Eds.; PMLR: 2019; Volume 97, pp. 1331–1340.
19. Giszter, S.F. Motor primitives—new data and future questions. *Curr. Opin. Neurobiol.* **2015**, *33*, 156–165.
20. Arie, H.; Arakaki, T.; Sugano, S.; Tani, J. Imitating others by composition of primitive actions: A neuro-dynamic model. *Robot. Auton. Syst.* **2012**, *60*, 729–741.
21. Riedmiller, M.; Hafner, R.; Lampe, T.; Neunert, M.; Degrave, J.; van de Wiele, T.; Mnih, V.; Heess, N.; Springenberg, J.T. Learning by Playing Solving Sparse Reward Tasks from Scratch. *arXiv* **2018**, arXiv:1802.10567.
22. Barto, A.G.; Konidaris, G.; Vigorito, C. Behavioral hierarchy: exploration and representation. In *Computational and Robotic Models of the Hierarchical Organization of Behavior*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 13–46.
23. Konidaris, G.; Barto, A.G. Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining. *Adv. Neural Inf. Process. Syst. (NIPS)* **2009**, *22*, 1015–1023.
24. Barto, A.G.; Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discret. Event Dyn. Syst.* **2003**, *13*, 41–77.
25. Manoury, A.; Nguyen, S.M.; Buche, C. Hierarchical affordance discovery using intrinsic motivation. In Proceedings of the 7th International Conference on Human-Agent Interaction; ACM: Kyoto, Japan, 2019; pp. 196–193.
26. Kulkarni, T.D.; Narasimhan, K.; Saeedi, A.; Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3675–3683.

27. Duminy, N.; Nguyen, S.M.; Duhaut, D. Learning a set of interrelated tasks by using sequences of motor policies for a strategic intrinsically motivated learner. In Proceedings of the 2018 Second IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, USA, 31 January–2 February 2018.
28. Schaal, S. Learning from demonstration. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1997; pp. 1040–1046.
29. Billard, A.; Calinon, S.; Dillmann, R.; Schaal, S. *Handbook of Robotics*; Number 59; Chapter Robot Programming by Demonstration; MIT Press: Cambridge, MA, USA, 2007.
30. Muelling, K.; Kober, J.; Peters, J. Learning table tennis with a mixture of motor primitives. In Proceedings of the 2010 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 6–8 December 2010; pp. 411–416.
31. Reinhart, R.F. Autonomous exploration of motor skills by skill babbling. *Auton. Robot.* **2017**, *41*, 1521–1537.
32. Taylor, M.E.; Suay, H.B.; Chernova, S. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*;International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 2011; pp. 617–624.
33. Thomaz, A.L.; Breazeal, C. Experiments in Socially Guided Exploration: Lessons learned in building robots that learn with and without human teachers. *Connect. Sci.* **2008**, *20*, 91–110.
34. Grollman, D.H.; Jenkins, O.C. Incremental learning of subtasks from unsegmented demonstration. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 261–266.
35. Duminy, N.; Nguyen, S.M.; Duhaut, D. Learning a Set of Interrelated Tasks by Using a Succession of Motor Policies for a Socially Guided Intrinsically Motivated Learner. *Front. Neurorobot.* **2019**, *12*, 87.
36. Argall, B.D.; Browning, B.; Veloso, M. Learning robot motion control with demonstration and advice-operators. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 399–404.
37. Chernova, S.; Veloso, M. Interactive Policy Learning through Confidence-Based Autonomy. *J. Artif. Intell. Res.* **2009**, *34*, 1.
38. Nguyen, S.M.; Oudeyer, P.Y. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn J. Behav. Robot.* **2012**, *3*, 136–146.
39. Cakmak, M.; Chao, C.; Thomaz, A.L. Designing interactions for robot active learners. *IEEE Trans. Auton. Ment. Dev.* **2010**, *2*, 108–118.
40. Begus, K.; Southgate, V. *Active Learning from Infancy to Childhood*; Springer: Berlin/Heidelberg, Germany, 2018; Chapter Curious Learners: How Infants' Motivation to Learn Shapes and Is Shaped by Infants' Interactions with the Social World; pp. 13–37.
41. Poulin-Dubois, D.; Brooker, I.; Polonia, A. Infants prefer to imitate a reliable person. *Infant Behav. Dev.* **2011**, *34*, 303–309. doi:10.1016/j.infbeh.2011.01.006.
42. Fournier, P.; Colas, C.; Sigaud, O.; Chetouani, M. CLIC: Curriculum Learning and Imitation for object Control in non-rewarding environments. *IEEE Trans. Cogn. Dev. Syst.* **2019**, 1, doi:10.1109/TCDS.2019.2933371.
43. Duminy, N.; Nguyen, S.M.; Duhaut, D. Effects of social guidance on a robot learning sequences of policies in hierarchical learning. In Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018.
44. Asada, M.; Hosoda, K.; Kuniyoshi, Y.; Ishiguro, H.; Inui, T.; Yoshikawa, Y.; Ogino, M.; Yoshida, C. Cognitive developmental robotics: A survey. *IEEE Trans. Auton. Ment. Dev.* **2009**, *1*, 12–34.
45. Cangelosi, A.; Schlesinger, M. *Developmental Robotics: From Babies to Robots*; MIT Press: Cambridge, MA, USA,2015.
46. Nguyen, S.M.; Oudeyer, P.Y. Socially Guided Intrinsic Motivation for Robot Learning of Motor Skills. *Auton. Robot.* **2014**, *36*, 273–294.
47. Kubicki, S.; Pasco, D.; Hoareau, C.; Arnaud, I. Using a tangible interactive tabletop to learn at school: empirical studies in the wild. In *Actes de la 28ième conférence francophone sur l'Interaction Homme-Machine*; ACM: New York, NY, USA, 2016; pp. 155–166.
48. Pastor, P.; Hoffmann, H.; Asfour, T.; Schaal, S. Learning and generalization of motor skills by learning from demonstration. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 763–768.
49. Stulp, F.; Schaal, S. Hierarchical reinforcement learning with movement primitives. In Proceedings of the 2011 11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, 26–28 October 2011; pp. 231–238.
50. Da Silva, B.; Konidaris, G.; Barto, A.G. Learning Parameterized Skills. *arXiv* **2012**, arXiv:1206.6398.
51. Sutton, R.S.; Precup, D.; Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **1999**, *112*, 181–211.
52. Vygotsky, L.S. *Mind in Society: the development of higher psychological processes*; Cole, M., John-Steiner, V., Scribner, S., Souberman, E., Eds.; Harvard University Press, Cambridge, MA: 1978.
53. Nguyen, S.M.; Oudeyer, P.Y. Interactive Learning Gives the Tempo to an Intrinsically Motivated Robot Learner. In Proceedings of the 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, 29 November–1 December 2012.