CHIME: an Adaptive Hierarchical Representation for Continuous Intrinsically Motivated Exploration

Alexandre Manoury[1,3] Sao Mai Nguyen[1,3] Cédric Buche[2,3]

***Abstract*—For a life-long learning, robots need to learn and adapt to the environment to complete multiple tasks ranging from low-level to high-level tasks, using simple actions or complex ones. Current intrinsically motivated solutions often rely on fixed representations of this environment to define possible tasks, limiting the possibility to adapt to new or changing ones. We propose an algorithm that is able to autonomously 1) self-discover tasks to learn in its environment 2) discover the relationship between tasks to leverage its acquired knowledge on low-level tasks to solve high-level tasks 3) devise a sequence of policies of unbounded length to complete the tasks. Our algorithm, named Continual Hierarchical Intrinsically Motivated Exploration (CHIME), uses planning to build chains of actions, the learning of a hierarchical representation of tasks to reuse low-level skills for high-level tasks and intrinsically-motivated goal babbling to discover new subtasks and orient its learning in its high-dimensional continuous environment. To highlight the features of CHIME, we implement it in a simulated mobile robot where it can move and place objects.**

***Index Terms*—Intrinsic motivation, Goal-babbling, Hierarchy, Planning, Adaptive, Continuous learning, Life-long learning**

I. Introduction

The capacity to adapt continuously to our environment is one of the key questions to understand the human development. Indeed, discovering how to interact with unknown elements in our surroundings while being able to generalise these competences to new tasks and to new objects is an essential part of being able to live and operate within it. This capacity of continuous learning, called life long learning, constitutes one of the main challenges a service robot has to tackle to operate in a real life environment.

We thus want to explore the possibility, for a robot, to discover and learn to perform various tasks from scratch in an unknown environment. With the perspective of a robot operating in a daily life environment, we want these tasks to be multiple, interrelated and of various complexity: from simply moving itself to using tools in order to reach inaccessible objects. To learn these tasks, the robot has to build a representation of the relations between the interrelated tasks.

II. Context

A. Active skill learning

Many approaches to motor skill learning have been proposed and studies learning forward and inverse models, mapping motor policies to sensorimotor outcomes in order to complete multiple learning tasks, are numerous [1], [2]. Still, as the sensorimotor and the task spaces increase in size and dimensionality, no comprehensive exploration is possible, and data collection faces the curse of dimensionality [3].

Some proposed approaches have been inspired by the human psychology and, more specifically, by how infants drive their exploration [4], [5]. Indeed, intrinsic motivation, or curiosity, has been pointed out to be a key factor of exploration [6].

Solutions even more inspired by teleological behaviours [7], using goal babbling, improve the robustness to high-dimensional policy spaces [8]. Such as the intrinsically motivated SAGG-RIAC algorithm [9] However, as the number of tasks or the dimension of the task space increases, these methods become less efficient [10], [11].

B. Planning sequences of policies

Considering daily environments, a robot may have to perform sequences of policies to complete high-level tasks. Approaches learning such sequences have been made, through procedures [12], or by combining Q-learning and intrinsic motivation [13]. Other approaches use options, that enables temporally abstract knowledge to be included in the reinforcement learning framework [14].

C. Hierarchical tasks representation learning

Additionally, infants seem to naturally focus on gradually difficult tasks [15]. They first learn simple sensorimotor control, then build upon it to learn more complex tasks.

As shown in [16] and [17] intrinsic motivation helps guiding the exploration even in an hierarchical architecture, with more numerous models but each being more simple to learn. [5] proposes a robotic framework based upon this to learn models using a given hierarchical representation.

D. Adaptation of the task hierarchy

To avoid having to specify this hierarchy by hand, methods have been proposed to learn this hierarchy: [12] learns it using a pre determined set of task subspaces for instance.

We extend this approach to let the robot infer by itself and modify its hierarchical representation. The goal is to fit in the best possible way to the environment, even if a change occurs within. We use a similar idea to the adaptive SVM update proposed in [18]: features improving precision are added to the SVM and others removed. In our case this means deciding what models to learn and on which spaces.

Using intrinsic motivation, planning and a task hierarchy, we propose in this paper the algorithm Continual Hierarchical

[1] IMT Atlantique, Brest, France. alexandre.manoury@gmail.com
[2] ENIB, Brest, France.
[3] Lab-STICC, CNRS

Intrinsically Motivated Exploration (CHIME), an extension of SAGG-RIAC, to learn multiple hierarchically organized tasks in high dimension spaces, and adapt to new environments.

III. Approach

We wish for a robot capable of learning its hierarchical representation of the environment. Our approach is grounded in developmental psychology, using intrinsic motivation to decide what to explore autonomously.

A. Problem formalization

Let us consider a robot that can interact with its environment by sequences of motions of unbounded length.

Each of these motions is a primitive policy: $\pi \in \Pi \subset \mathbb{R}^N$. As the policies executed may be of various kinds and dimensions depending on the actuators involved, we partition Π into different subspaces $\Pi_i \subset \Pi$. Π_i is called a primitive policy space and $\pi_i \in \Pi_i$ corresponds to a motor command that can be sent to one or several actuators of the robot. E.g. the angle value for a servo motor.

The robot can also perform sequences of primitive policies of any length $n, \pi = [\pi_1, \dots, \pi_n] \in \Pi^n$. The policy space available for the robot is thus $\Pi^{\mathbb{N}} \subset \mathbb{R}^{\mathbb{N}}$.

Each of the policies performed in the environment may have consequences observable by the robot, we call such consequences observations and note them $\omega \in \Omega \subset \mathbb{R}^M$. And again, as each observation may be of various kinds, we partition Ω into different subspaces $\Omega_i \subset \Omega$. Ω_i is called an observable space. E.g. an object position.

Let us note $\mathcal{F} = \Pi \cup \Omega$ the feature ensemble, grouping both primitive policies and observations. Thus, $\mathcal{F} = \{\Pi_1, \dots, \Pi_N, \Omega_1, \dots, \Omega_M\}$ and we note $\mathcal{F}_i$, called feature space, each one of its subspaces.

To learn how to interact with its environment, the robot learns models of relations between policies π and observations ω, and more generally between features f and observations ω. Indeed, our robot may learn how to reach an observation value by inducing first a change in another observable of the environment. E.g. pushing an object can be performed after placing the robot close to the object.

Let us note a model $M(\Omega_A \to \mathcal{F}_A) : \omega_g \mapsto \pi'_f$. ω_g is an outcome we want to induce in the environment, called goal, and π'_f is the feature policy, or policy, to reach or execute (if primitive or not) in order to reach ω_g. Please note that a policy can be an observable to reach in order to produce ω_g, and may not be a primitive policy. A feature policy is noted π'. Ω_A can be a single observable space or a union of them: $\Omega_A = \cup_{i \in A = \{\dots\}} \Omega_i$. Likewise, $\mathcal{F}_A$ is a single feature space or an union of them: $\mathcal{F}_A = \cup_{i \in B = \{\dots\}} \mathcal{F}_i$. M can then be used as an inverse model: indicating which π'_f should be executed in order to reach best a given ω_g observable goal, based on the data accumulated so far in the robot dataset. Let us note $\mathcal{D}$ this dataset, where all the exploration data are to be stored, and L_M the application realizing this prediction.

Each model M can be seen as a basic skill, letting the robot perform a given task. E.g. reaching a a specific position.

Let us note $\mathcal{H}$ the ensemble of the models used by our robot. As our robot aimed to be adaptive, $\mathcal{H}$ varies along time.

B. Environmental setup

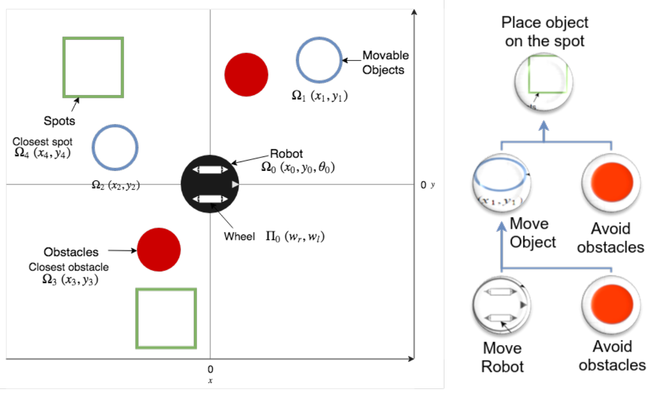


Fig. 1: Experimental setup on the left, presenting the primitive policy space Π_0 and the observable spaces Ω_k. On the right, an example of a hierarchical task possible in this setup.

To help describing CHIME, we first present our experimental setup: let us consider a robot operating in a 2D room containing 2 movable objects and 2 fixed ones, as represented in Figure 1. Additionally, spots are present on the floor of the room, they are not solid and placing objects on them will modify an environmental observable value r_{spots}. This setup is designed to learn interraled tasks of incresing difficulty.

The only primitive policy space considered for this two-wheeled robot is a 2 dimension parameter controlling the two wheels : $\Pi = \Pi_{Wheels} = [-1, 1]^2$.

The other spaces are all observable spaces: Ω_i correspond, in the order, to the robot position, an object position (relative to the robot position), the nearest obstacle position, the nearest spot position and a value rewarding placing objects on spots.

C. Algorithmic architecture

CHIME is iterative and learns by episodes, as outlined in Figure 2. At each iteration a primitive policy $\pi \in \Pi$ is executed and the observations $\omega_i \in \Omega_i$ are collected and processed.

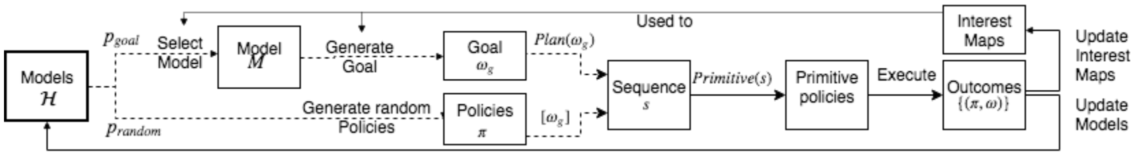
The robot starts its exploration without any a-priori knowledge on the hierarchy to learn. Thus, starting with $\mathcal{D}, \mathcal{H} = \emptyset$.

When starting an episode (left of the figure), the robot possesses two exploration methods (represented by dashed arrows): either a goal oriented exploration or a random policy one. Both methods select, respectively according to its interest model or by random, a sequence of features $s = [\pi'_k, k \in [1, n]] \in \mathcal{F}^n$ to be executed.

In the first case, the robot first stochastically selects a model $M \in \mathcal{H}$ in respect to the interest of each model $Interest(M)$. $Interest(M)$ represents the interest the robot considers to have in exploring such model. It is later detailed in III-D. Once M is selected, the robot selects a goal $\omega_g \in \Omega$ based on the interest measure, as described in III-D. It then tries to reach ω_g during one or several iterations. At each iteration it uses planning and its models to compute a sequence of feature subgoals $s = Plan(\omega_g) \in \Pi^n$ to reach ω_g. We note $Plan : \Omega \to \Omega^n, \omega \mapsto s$ the application constructing such a sequence. In our implementation, $Plan$ uses the RRT algorithm to plan this sequence.

When choosing the random policy exploration, the robot selects a random feature policy space $\mathcal{F}_i$ and a random policy

Fig. 2: Global architecture of CHIME. Dashed arrows correspond to ϵ-greedy alternative paths, with the probabilities p_1 and p_2. An episode starts on the left, by selecting a Model M. The different functions of this architecture are explained in the following sections.



within $\pi'_r \in \mathcal{F}_i$. π'_r can either be a primitive policy, and thus be executed directly, or an observable goal that needs to be transformed into primitive policies first. To match with the previous case we define the sequence $s = [\pi'_r]$ to be executed: if $\pi'_r \in \Omega$, this is a sequence of subgoals. If $\pi'_r \in \Pi$, this is directly a sequence of primitive policies.

In both cases, we obtain a sequence of feature policies $s = [\pi'_k, k \in [1,n]] \in \mathcal{F}^n$ that needs to be executed in the environment. For each π'_k, π'_k may currently not be a primitive policy, and thus not executable, so we first break it down into a sequence of primitive policies using $Primitive$: $\pi_k = Primitive(\pi'_k) \in \Pi^n$. Let us note $Primitive : \Omega \rightarrow \Pi^n, f \mapsto \pi$ the application constructing such a sequence. Once π_k is obtained, we execute it and then retrieve the observations $\omega \in \Omega$ from the environment and store them in $\mathcal{D}$.

Once the policy has been executed, the outcome observable has been retrieved and stored into $\mathcal{D}$, a mechanism then uses the data to update the $Interest$ values and another to update $\mathcal{H}$ by creating or modifying models, as described in III-F.

D. Intrinsic motivation

To decide what model and goal to select, CHIME uses intrinsic motivation, letting the robot decide autonomously what to explore. For this part, we operate similarly to the SAGG-RIAC algorithm [9].

The intrinsic motivation revolves around a notion of interest, indicating which region of Ω are interesting to explore. This notion is computed using a progress value $Progress^i_M(\omega_i) = |Competence^{i-1}_M(\omega_i) - Competence^i_M(\omega_i)|$ for each new observable ω_i, corresponding to the difference of competence before and after adding ω_i to $\mathcal{D}$. $Competence_M : \Omega \rightarrow [0,1]$ is a measure of competence of how the robot is confident when computing a policy prediction $L_M(\omega_i)$ using the inverse model M. It then computes $Interest_i(\omega_i) = \frac{Progress^t_M(\omega_i)}{dt}(i)$, the derivative of the progress of ω_i using recents items from $\mathcal{D}$. Let us note $Competence(M)$ the competence of M for all the data in $\mathcal{D}$.

The whole mechanism is described in [9].

E. Hierarchical representation

As $s = [s_0, \ldots, s_n] \in \Omega^n$, it first needs to be converted to a sequence of primitive policies $\pi \in \Pi^m$ in order to be executable. This is done by using the learned inverse models on each element of s. We then get $\pi'_1 = L_M(s) = [L_M(s_k), s_k \in s] \in \mathcal{F}^n$.

As the tasks are hierarchically organised, π'_1 may be a sequence of feature policies to be reached in order to complete our high-level task ω_g. The robot have to apply a new phase of planning and inverse model prediction, using a model M_2, in order to obtain a lower level sequence $\pi'_2 = L_{M2}(plan(\pi'_1)) \in \mathcal{F}^m_2$. If π'_2 is a primitive policy sequence, it can then be executed without further work, else the process starts again with π'_2 until obtaining a primitive policy sequence.

F. Adaptive hierarchical representation

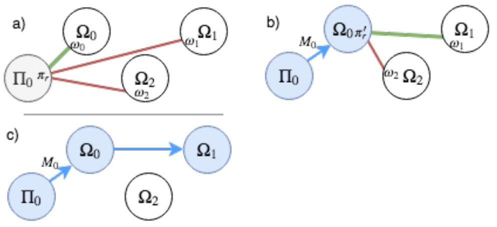


Fig. 3: Evolution of environment representation. From (a) to (c) the system has decided to create different models (represented by blue directed edges) between spaces (represented by nodes). The green and red edges show the correlation between the policies performed π_r and the outcomes ω_k obtained in the different spaces.

CHIME uses an adaptive structuring of its environment to let the agent create its own hierarchical representation as illustrated in Figure 3.

Algorithm 1 explicits the update procedure. At each learning episode, it decides what models to create (l.2), when to add a feature to a model (l.4) or to remove one (l.6). β is a threshold avoiding the creation of uncorrelated models, and ϵ avoiding flapping between addition/deletion of $\mathcal{F}_{new}$ in a model.

Figure 3 shows an example of a model evolution during a learning session while using this dynamic model structuring method. It first performs random policies on $Wheels$ (Fig 3 a), measures its correlation with the observations in the different observable spaces and selects $RobotPosition$. It then creates $M(\Omega_{RobotPos} \rightarrow \Pi_{Wheels})$ (Fig 3 b). Similarly, it creates $M(\Omega_{ObjectPos} \rightarrow \Pi_{RobotPos})$ (Fig 3 c).

IV. Experimental results

We have tested CHIME in a python-simulated environment presenting hierarchical tasks to be learned and obstacles to be avoided, as described in III-B.

We evaluated the results in regards to the hierarchical representation built by the robot compared to our ground truth and, secondly, to the evaluation of task competence, using the mean of $Competence_M(p)$ over a predefined testbench.

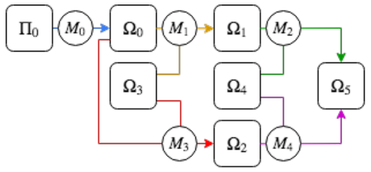
We use the setup presented in III-B to which we add $\Pi_k \subset \mathbb{R}, k \in [1,9]$, 9 primitive policy spaces producing no effect in the environment, like in the previous setup. We also add $\Omega_k \subset \mathbb{R}, k \in [6,99]$, irrelevant observable spaces made of white noise or fixed values. These additions are useful to verify that CHIME only creates relevant models.

We did 10 runs and obtained the hierarchical representation presented in Figure 4 (top). In every run, CHIME has been able

Algorithm 1 The *updateModel* function

```
1: if $M(\Omega_u \to any) \notin \mathcal{H}$ and $Competence(M(\Omega_u \to \mathcal{F}_u)) \geq \beta$ then
2:     $H \leftarrow M(\Omega_u \to \mathcal{F}_u)$
3: for each $M(\Omega_M \to \mathcal{F}_M) \in \mathcal{H}$, and for each $\mathcal{F}_{new} \in E$ do
4:     if $\mathcal{F}_{new} \not\subset \mathcal{F}_M$ and $Competence(M(\Omega_M \to \mathcal{F}_M \cup \mathcal{F}_{new})) \leq Competence(M(\Omega_M \to \mathcal{F}_M)) - \epsilon$ then
5:         $\mathcal{F}_M \leftarrow \mathcal{F}_M \cup \mathcal{F}_{new}$
6:     if $\mathcal{F}_{remove} \subset \mathcal{F}_M$ and $Competence(M(\Omega_M \to \mathcal{F}_M \setminus \mathcal{F}_{remove})) \leq Competence(M(\Omega_M \to \mathcal{F}_M)) - \epsilon$ then
7:         $\mathcal{F}_M \leftarrow \mathcal{F}_M \setminus \mathcal{F}_{remove}$
```

to discover the environment hierarchy and to avoid irrelevant spaces, requiring around 2300 ($\sigma^2 = 800$) iterations. It has successfully identified that using the wheels Π_0 is required to move the robot Ω_0, that moving the robot Ω_0 and considering the obstacles Ω_3 is pertinent in order to move the objects $\Omega_{1,2}$, and finally, that moving either the object 1 or 2 and considering the spot position Ω_4 can lead to control the reward value Ω_5.



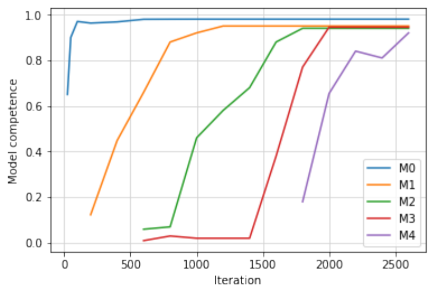


Fig. 4: Model hierarchy discovered (top). Competence evaluation (bottom) of each model M along the exploration iterations. When no competence is present, the model has just not been added yet.

On Figure 4 (bottom) it is visible that all the models have a good final competence. This is coherent as no noise is present in this setup, the only final competence errors are due to approximations or imperfections in the physics engine. We can also see on this figure the hierarchical adaptive aspect of CHIME: low level models and tasks (as Ω_0) are learned at the very beginning, and once mastered, they let more higher level models to be learned on top of them.

When applying SAGG-RIAC to this setup, only the first model is learned as it cannot handle hierarchical models.

To be concise, we didn't present other results, but adding or removing possible tasks in the setup are also natively well managed and discovered by CHIME.

Apart from the representation building, this setup also shows the capacity of CHIME to tackle hierarchical interrelated tasks, using its planning and interrelated learning of tasks.

V. Conclusion and Future Works

For multi-task learning, we have presented CHIME, to learn how to complete high-level tasks using unbounded sequences of policies. The learning process allows the learner to 1) discover autonomously subtasks to build upon for high-level tasks 2) discover the relationship between tasks and 3) devise a sequence of policies of unbounded length and complexity to complete this task. We have shown that an intrinsically motivated algorithm can learn relevant spaces and build hierarchical models to address multitask learning for a wide variety of tasks including low-level and high-level tasks.

In future works, we want to deepen the analysis of the behaviour of CHIME in different contexts, using more complex experimental setups and comparing to more algorithms.

References

[1] B.A. Francis and W.M. Wonham. The internal model principle of control theory. *Automatica*, 12(5):457 – 465, 1976.
[2] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317 – 1329, 1998.
[3] R. Bellman. *Dynamic programming*. 1957.
[4] P.-Y. Oudeyer and F. Kaplan. Intelligent Adaptive Curiosity: a source of Self-Development. *Science*, 117:127–130, 2004.
[5] S. Forestier and Oudeyer P.-Y. Overlapping Waves in Tool Use Development: a Curiosity-Driven Computational Model. *IEEE International Conference Developmental Learning and Epigenetic Robotics*, pages 1859–1864, 2016.
[6] Karen A. Miller, Edward L. Deci, and Richard M. Ryan. Intrinsic Motivation and Self-Determination in Human Behavior. *Contemporary Sociology*, 17(2):253, 1988.
[7] G. Csibra and G. Gergely. Obsessed with goals: Functions and mechanisms of teleological interpretation of actions in humans. *Acta Psychologica*, 124(1):60 – 78, 2007.
[8] R. F. Reinhart. Autonomous exploration of motor skills by skill babbling. *Autonomous Robots*, 41(7):1521–1537, 2017.
[9] A. Baranes and P.-Y. Oudeyer. *Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots*. 2013.
[10] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.
[11] S. M. Nguyen and P.-Y. Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, 2014.
[12] N. Duminy, S. M. Nguyen, and D. Duhaut. Learning a set of interrelated tasks by using sequences of motor policies for a strategic intrinsically motivated learner. *IEEE International on Robotic Computing*, pages 288–291, 2018.
[13] C. Craye, D. Filliat, and J.-F. Goudou. Environment Exploration for Object-Based Visual Saliency Learning. *International Conference on Robotics and Automation*, pages 2303–2309, May 2016.
[14] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181 – 211, 1999.
[15] J. Gottlieb, P.-Y. Oudeyer, M. Lopes, and A. Baranes. Information-seeking, curiosity, and attention: Computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585–593, 2013.
[16] A. G. Barto. Intrinsically motivated learning of hierarchical collections of skills. *Development and Learning*, pages 112–119, 2004.
[17] C.M. Vigorito and A.G. Barto. Intrinsically Motivated Hierarchical Skill Learning in Structured Environments. *IEEE Transactions on Autonomous Mental Development*, 2(2):132–143, 2010.
[18] E. Ugur and J. Piater. Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. pages 1–13, 2016.