Research Article Open Access

George Velentzas, Theodore Tsitsimis, Iñaki Rañó, Costas Tzafestas, and Mehdi Khamassi*

Adaptive reinforcement learning with active state-specific exploration for engagement maximization during simulated child-robot interaction

https://doi.org/10.1515/pjbr-2018-0016
Received February 2, 2018; accepted June 29, 2018

Abstract: Using assistive robots for educational applications requires robots to be able to adapt their behavior specifically for each child with whom they interact. Among relevant signals, non-verbal cues such as the child's gaze can provide the robot with important information about the child's current engagement in the task, and whether the robot should continue its current behavior or not. Here we propose a reinforcement learning algorithm extended with active state-specific exploration and show its applicability to child engagement maximization as well as more classical tasks such as maze navigation. We first demonstrate its adaptive nature on a continuous maze problem as an enhancement of the classic grid world. There, parameterized actions enable the agent to learn single moves until the end of a corridor, similarly to "options" but without explicit hierarchical representations. We then apply the algorithm to a series of simulated scenarios, such as an extended Tower of Hanoi where the robot should find the appropriate speed of movement for the interacting child, and to a pointing task where the robot should find the child-specific appropriate level of expressivity of action. We show that the algorithm enables to cope with both global and local non-stationarities in the state space while preserving a stable behavior in other stationary portions of the state space. Altogether, these results suggest a promising way to enable robot learning based on non-verbal cues and the high degree of non-stationarities that can occur during interaction with children.

Keywords: human-robot interaction, reinforcement learning, active exploration, meta-learning, autonomous robotics, engagement, joint action

1 Introduction

Among the large set of possible domains of applications of assistive robotics, a substantial subset requires social interaction between the human and the robot, and in particular the ability of the robot to adapt to non-verbal social signals [1]. For example, a robot assisting elderly people at home during daily life should be able to detect when the human has difficulties reaching a particular object or even standing up and moving, in order to spontaneously and quickly come to help. Another example relates to educational applications where a small humanoid robot can assist a human teacher, the robot being here considered as an educational tool, in order to promote typical or Autistic Spectrum Disorders (ASD) children's interest in educative games, and help further develop their social skills [2–4] and enhance their motivation [5].

In the case of assistive robots for educational applications, one particularly important non-verbal social signal that has received increasing interest in the last few years is the notion of maximizing a child's engagement in the game or task [6–8]. Mutual engagement can be defined as "the process by which interactors start, maintain and end their perceived connection to each other during an interaction" [9]. This can give crucial information about the degree with which the child is involved in joint attention and joint action with the other social agents. A number of different measures have been considered as relevant to estimate a human's engagement during social interaction, among which body posture and gaze [6, 10]. While proposing a full model of human engagement is a complex task and is out of the scope of the present work, the goal here is

George Velentzas, Theodore Tsitsimis, Costas Tzafestas: School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece; Email: ktzaf@cs.ntua.gr

Iñaki Rañó: Intelligent Systems Research Centre, Ulster University, UK; E-mail: i.rano@ulster.ac.uk

***Corresponding Author: Mehdi Khamassi:** Sorbonne Université, CNRS, Institute of Intelligent Systems and Robotics, F-75005 Paris, France; E-mail: mehdi.khamassi@upmc.fr

to enable a robot to learn to efficiently and rapidly adapt its behavior in response to changes in a child's body posture and gaze in order to improve the efficiency of the educational game in terms of development of the child's social skills.

Attempting to develop robot learning abilities in this social context emphasizes two important challenges among those that are common to non-social applications: (1) the choice to learn either discrete or continuous actions that should be performed by the robot in order to produce a sufficiently fine-grained behavior but nevertheless categorizable and understandable by the child during interaction; (2) the choice of the level of exploration in learning that is appropriate to cope with the high level of non-stationarity that can occur during social interaction. Previous researches have applied reinforcement learning to discrete action spaces, including for human-robot interaction applications (*e.g.*, see [11]). Nevertheless, the decomposition of the task into a small set of discrete actions requires important prior human knowledge and may prevent generalization to more complex tasks requiring continuous motor actions. Alternatively, applications of reinforcement learning to continuous action spaces [12, 13] are promising for fine-grained robot behavioral adaptation in real-world applications (see [14] for a recent review). Nevertheless, important human knowledge is still required here (*e.g.* by demonstration) to reduce exploration to small relevant portions of the large continuous action space.

In [15, 16], we have previously proposed to apply the framework of Parameterized Action Space Markov Decision Processes (PAMDP) [17, 18] to human-robot interaction because it constitutes a promising intermediate solution between discrete and continuous action learning. This framework permits rich behavioral repertoires by enabling a robot to learn to choose between a small set of discrete actions (*e.g.*, shooting in a ball, turning, running) and at the same time to learn continuous parameters of these actions (*e.g.*, shooting strength, rotation angle, running speed). Moreover, in order to apply this framework to tasks with non-stationarities, we introduced active exploration principles [19–23] to avoid employing a fixed predetermined balance between exploration and exploitation. Nevertheless, we previously focused on single state scenarios where a single action by the robot is sufficient to trigger an outcome. This permitted a proof-of-concept of the method and analyses of robustness to uncertainty and perturbations in human engagement measures during interaction [16].

Here we extend this framework to multiple states scenarios – requiring sequences of actions to complete the task – in order to permit its generalization to a variety of educational games that can be used during child-robot interaction. One important novelty here is to propose a state-specific active exploration process, which permits to cope with local non-stationarities in the state space without interfering with learning in other stable portions of the state space. In addition, because there may be several continuous parameters of actions that could be modulated online (*e.g.*, duration, velocity, gaze orientation, etc.), we have extended the algorithm to multiple continuous parameters per action. We present a series of validation experiments to evaluate the algorithm in simulation. We first test it in a navigation task in a continuous maze and show that it can cope with sequences of abrupt non-stationarities such as changes in the maze topology as well as in the goal location. We then test the algorithm on simulated child-robot interaction experiments and show that it outperforms the previous version with non-state-specific active exploration, and can cope with local and global change-points, as well as with deterministic and stochastic tasks. We then perform a Tower of Hanoi experiment where in addition to achieving the task within the minimum number of steps the algorithm has also to learn to build the tower with the appropriate speed of movement for the child with whom it interacts. Finally, we simulate a task inspired by a pilot experiment that we made with ASD children where the robot needs to find the child-specific appropriate level of expressivity of action while pointing at an object in order to make the child react to help the robot reach the object.

Figure 1: Setup used for the pilot real child-robot interaction experiment.

2 Methods

2.1 General experimental paradigm

The general experimental paradigm adopted here consists in having a small humanoid robot interact with children (one at a time), under the supervision of an observing human adult, and finding the appropriate robot behavior to maximize children's engagement in the task. This paradigm follows the objectives defined in the framework of the EU-funded project BabyRobot (H2020-ICT-24-2015-6878310), where a set of child-robot interaction use-cases have been designed and implemented to study the development of specific socio-affective, communication and collaborative skills in typical and ASD children. In this framework, we have set up a pilot experiment where the NAO robot is interacting with a child (Figure 1), and repeatedly points at an unreachable object while varying the level of expressivity of its pointing gesture (*i.e.*, opening-and-closing hand for a certain duration, bending its torso with a certain angle in the direction of the object, gazing at the child for a certain duration) until the child understands the "intention" of the robot and engages herself into joint action in order to help the robot grasp the object. Importantly, because different children may not like the same level of robot action expressivity and mutual gaze, especially in the case of ASD children, we want the robot to learn the appropriate level of expressivity specifically for each interacting child. Moreover, because the child's preferences and attention may not be stationary, resulting in drops of engagement after long periods with a certain level of expressivity, we also want the robot to dynamically adapt its behavior to variations of the child's engagement. Finally, in a second pilot scenario, the child and the robot successively perform a Tower of Hanoi task while being observed and helped by the other. In this case, the task involves a sequence of states where the objects are at different locations. The goal of the scenario is to develop the child's social skills by making her understand when the robot has difficulties solving the task and thus may need help.

Here we present extensions of our previous active exploration reinforcement learning algorithm [15, 16] to cope with these scenarios requiring multiple continuous parameters per action and multiple task states. We then present a series of numerical simulations to show that it can solve these tasks. The first important idea here is that active exploration will be a way for the algorithm to track variations of the child's engagement (transformed into a social reward signal) in order to re-explore each time a consistent change is detected and thus adapt faster. The second important thing to stress is that the algorithm learns in parallel a discrete Markov Decision Process (MDP) to solve these tasks (which can be seen as a non-social reward) and continuous parameters of action (*i.e.*, expressivity, speed of movement) in order to maximize child engagement during the task (which can be seen as a social reward).

2.2 State-specific exploration

Exploration during a learning procedure should be handled in a sophisticated manner in order to find a decision strategy that does not suffer from initialization biases, and avoids converging to local-minima in terms of optimality. A commonly used strategy known as "annealing" is to monotonically decrease exploration in time. However, in dynamic environments this approach will fail as the distributions of rewards and the state transition function might be time-dependent. In particular, when the environmental dynamics are not constrained by inertial rules (*i.e.*, they are not traceable), the decision agent should incorporate intrinsic characteristics of adaptivity. In [24] a biologically plausible method was proposed to adaptively tune the exploration levels as also the learning rate and the discount of rewards in a reinforcement learning framework. In [16] this idea was expanded to parameterized action spaces, by tuning the uncertainty of the continuous action parameters in parallel to dynamically regulating exploratory choices between discrete actions (*e.g.*, choosing between pointing at the red cube or the blue cube), and at the same time exploring the continuous parameters of expressivity or speed of execution of these discrete actions. This approach can be seen as a way to generalize when a hierarchical breakdown of a high dimensional continuous action space can result in having discrete subsets (viewed as the discrete actions), reducing the dimensionality of the computationally expensive continuous search to smaller regions of the action space.

Nevertheless, this previous work focused on single-state tasks where a single action is sufficient to produce an outcome (*i.e.*, a reward signal as a function of the change in the engagement of the human interacting with the robot). Here, we want to further generalize the approach to multi-state tasks where the robot shall perform a series of actions before completing the task. In such a case, the volatility of the state space may vary depending on the region. Some regions of the state space may produce stochastic transitions and rewards while others may be deterministic. Moreover, some regions may be stationary while others are volatile. As a simplistic example, let us consider a sce-

nario where a teacher wants to teach the mathematical operations of addition, subtraction, multiplication and division to students, having a redundant number of paradigms in mind. The instructor can choose different time allocations for the demonstration of each one. He/she can also tune the expressiveness of his/her explanations. The objective would be to complete the curriculum in certain time frames by also maximizing the students' engagement and understanding of the material. An instructor with many years of experience will have probably optimized the way addition is being taught. However, the instructor might be more uncertain about division, and may try different strategies for each class each year by using an estimation of students' engagement as a feedback. Going back to the general idea, using a global exploration level would result in an over-pessimistic approach, which could prevent the optimality of actions in states where there is no need for re-exploring.

Taking into consideration the above, here we expand the ideas from [16] by incorporating state-specific exploration strategy. We substitute the mid-term and long-term reward running-averages (which are not state-specific but global) of a state respectively with the state value $V(s)$ and $\bar{V}(s)$ as a running average of $V(s)$. More precisely, when the agent is in some state s, and performs a discrete action a with parameter values $\boldsymbol{\theta}$, observes a reward r and a resulting state $s^{'}$, we update $V(s)$, then update $\bar{V}(s)$ with $\bar{V}(s) \leftarrow \bar{V}(s) + \alpha_V(V(s) - \bar{V}(s))$ and use their difference $\bar{\delta}_V = V(s) - \bar{V}(s)$ to tune the exploration levels specifically for state s. This approach follows a common reasoning, backed up with neurobiological data described in [25], when the average of the returned rewards is greater than the average of the averages, then the current strategy is more promising and hence exploitation may be increased. Re-engaging exploration in an adaptive manner is a challenging problem since a "positive feedback" loop may result in an unstable strategy. To make it clear, even small incorrect increments of exploration may result in having a worse performance than the existing one. However, the observation of a degraded performance would be further regarded as an evidence for the need of an additional increment. Moreover, as described in [26], cases where a non-optimal action becomes the most promising one without a change of the reward distribution of the current performing actions, may stay "hidden" and untracked. A good strategy could then be to lower-bound the exploration levels, with the sacrifice of not achieving a "no regret" performance in stationary cases [27].

With the use of the above notation, the value $\bar{\delta}_V$ will be used to tune the uncertainty of the action decision strategy in state s, but also the uncertainty of the estimated parameter values of the performed action in state s.

2.3 Description of the algorithm

We consider a finite discrete state space $S = \{s_1, s_2, \ldots, s_k\}$ where each state $s \in S$ is represented by an m-dimensional feature vector $\boldsymbol{\phi}(s)$. The parameterized action space can be described in general with the use of a finite discrete set $A_d = \{a_1, a_2, \ldots, a_n\}$ where each $a \in A_d$ denotes a discrete action. Following the parameterized reinforcement learning framework [17], each discrete action has a number of m_a continuous parameters and therefore can be described by an m_a-dimensional vector $\boldsymbol{\theta}^a \in \mathbb{R}^{m_a}$. Using the above notation the action space A is then written as

$$A = \bigcup_{a \in A_d} \{(a, \boldsymbol{\theta}^a) | \boldsymbol{\theta}^a \in \mathbb{R}^{m_a}\}$$

For learning the action values we will be using a simple Q-learning rule which can however be substituted with a different strategy depending on the task. Assume that the decision agent is in state s and chooses a discrete action a with a parameter vector $\boldsymbol{\theta}^a$ (we will later describe how), a reward r is returned and a transition to state $s^{'}$ takes place. With the observation of the quintuple $(s, a, \boldsymbol{\theta}^a, r, s^{'})$ the reward prediction error can be computed as

$$\delta_Q = r + \gamma \max_{a_j} Q(s^{'}, a_j) - Q(s, a)$$

where γ is the reward discount factor. Take note that the parameter vector $\boldsymbol{\theta}^a$ is not taken into consideration here, since the Q-values will be used as a measure for the evaluation of the discrete actions. However, we can follow a more general approach where the Q-values may not be directly accessible. With $\boldsymbol{\phi}(s)$ representing the m-dimensional feature vector, a linear function approximator can be used with this feature vector as an input to a neural network. Here for clarity we will be using a single layer network with m inputs and $n = |A_d|$ outputs (with $|\cdot|$ denoting the cardinality). Nevertheless the ideas can be expanded to deeper architectures. Let $\boldsymbol{W}$ be the $m \times n$ weight matrix of the network, where w_{ij} is the weight connecting the i-th component $\phi_i(s)$, with the j-th output, which corresponds to the Q-value of action a_j in state s. With this notation in mind it can be easily seen that

$$[\boldsymbol{W}^T \boldsymbol{\phi}(s)]_j = Q(s, a_j) \tag{1}$$

where $[\cdot]_j$ is the j-th element of a vector. With j being the index of the performed action such that $a = a_j$, the reward prediction error δ_Q is then

$$\delta_Q = r + \gamma \max_i [\boldsymbol{W}^T \boldsymbol{\phi}(s^{'})]_i - [\boldsymbol{W}^T \boldsymbol{\phi}(s)]_j \tag{2}$$

and for all $i \in \{1, 2, \ldots, m\}$ the weights w_{ij} are updated with the following rule

$$w_{ij} \leftarrow w_{ij} + \alpha_Q \delta_Q \phi_i(s) \tag{3}$$

where α_Q is a learning rate of choice.

The estimations of the parameter values for each action a can be represented by a matrix $\hat{\boldsymbol{\Theta}}^a$ of size $m_a \times k$, with $k = |S|$, where the j-th column of $\hat{\boldsymbol{\Theta}}^a$ will embody the current expectation $\hat{\boldsymbol{\theta}}^a = E[\boldsymbol{\theta}^a_\star | S = s_j]$, (*i.e.*, the approximation of the optimal parameter vector $\boldsymbol{\theta}^a_\star$ for action a in state s_j). In order to update the current expectation we will be using a continuous actor-critic algorithm [28]. At first, with $\boldsymbol{v}$ being the weight vector for the critic, the value function of each state s can be approximated as $\boldsymbol{v}^T\boldsymbol{\phi}(s)$ and the reward prediction error δ_V of the critic can be computed as

$$\delta_V = r + \gamma \boldsymbol{v}^T \boldsymbol{\phi}(s') - \boldsymbol{v}^T \boldsymbol{\phi}(s) \tag{4}$$

The full weight vector $\boldsymbol{v}$ for the value function can now be updated with the following step

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \alpha_C \delta_V \boldsymbol{\phi}(s) \tag{5}$$

To update the weights $\bar{\boldsymbol{v}}$ for the linear approximation of function $\bar{V}(s)$, the new output of the value function network for state s will be used as a target. With denoting the weights of the approximator as $\bar{\boldsymbol{v}}$ and using a learning rate α_V, the updates will then be

$$\bar{\boldsymbol{v}} \leftarrow \bar{\boldsymbol{v}} + \alpha_V (\boldsymbol{v}^T \boldsymbol{\phi}(s) - \bar{\boldsymbol{v}}^T \boldsymbol{\phi}(s)) \boldsymbol{\phi}(s) \tag{6}$$

And the error $\bar{\delta}_V = V(s) - \bar{V}(s)$ can then be computed as the difference of the two network outputs

$$\bar{\delta}_V = \boldsymbol{v}^T \boldsymbol{\phi}(s) - \bar{\boldsymbol{v}}^T \boldsymbol{\phi}(s) \tag{7}$$

For the approximation of $\hat{\boldsymbol{\Theta}}^a$, a network with weight matrix $\boldsymbol{G}^a$ of size $m \times m_a$ will be used for each action. Additionally, the values of the parameters may be constrained such that $\|\boldsymbol{\theta}^a\|_\infty \le \theta_{max}$. We consider this case as a general one, since the parameter search may be performed (and in fact this should be the case) in a symmetric space (regarding the boundary constraints) with a proper affine transform. With $\mathcal{F}_\theta(\cdot)$ being a piece-wise linear activation vector function, simply lower-truncating each component of the input at $-\theta_{max}$ and upper-truncating at $+\theta_{max}$. The estimation $\hat{\boldsymbol{\theta}}^a$ of the optimal parameter vector for action a in state s will therefore be $\mathcal{F}_\theta((\boldsymbol{G}^a)^T \boldsymbol{\phi}(s))$. With $\boldsymbol{\theta}^a$ being the current choice, we compute the displacement $\boldsymbol{e}$ as

$$\boldsymbol{e} = \boldsymbol{\theta}^a - \hat{\boldsymbol{\theta}}^a = \boldsymbol{\theta}^a - \mathcal{F}_\theta((\boldsymbol{G}^a)^T \boldsymbol{\phi}(s)) \tag{8}$$

Then for $i = \{1, \ldots, m\}$ and $j = \{1, \ldots, m_a\}$, the weights g^a_{ij} of $\boldsymbol{G}^a$ can be updated if $\delta_V > 0$ with

$$g^a_{ij} \leftarrow g^a_{ij} + \alpha_A \delta_V e_j \phi_i(s) \tag{9}$$

Take note that the updates for the estimators of the parameter values here are made according to δ_V as in the CACLA algorithm [28]. Going back to the beginning of the description, let us denote that the agent at each state $s \in S$ chooses an action-parameter tuple $(a, \boldsymbol{\theta}^a) \in A$ sampled from a joint probability distribution such that

$$p(a, \boldsymbol{\theta}^a | s) = P(a|s) p(\boldsymbol{\theta}^a | s, a)$$

For the discrete action choice we will be using the softmax layer as also done in [16], with the extension of using an adaptive state-specific inverse temperature $\beta(s)$ such that

$$P(a|s) = \frac{\exp(\beta(s) Q(s,a))}{\sum_a \exp(\beta(s) Q(s,a))} \tag{10}$$

where the Q-values are the outputs of the function approximator. The parameter vector is sampled from a multivariate Gaussian distribution centered at the current approximation $\hat{\boldsymbol{\theta}}^a = \mathcal{F}_\theta((\boldsymbol{G}^a)^T \boldsymbol{\phi}(s))$ of the optimal parameter vector $\boldsymbol{\theta}^a_\star$ with the use a diagonal covariance matrix $\boldsymbol{\Sigma}^a_s$ such that

$$p(\boldsymbol{\theta}^a | s, a) = \frac{1}{(2\pi)^{m_a/2} |\boldsymbol{\Sigma}^a_s|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{\theta}^a - \hat{\boldsymbol{\theta}}^a)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta}^a - \hat{\boldsymbol{\theta}}^a)} \tag{11}$$

The weight vectors for approximating $\beta(s)$ and the diagonal elements of the matrix $\boldsymbol{\Sigma}^a_s$ are updated right after the computation of $\bar{\delta}_V$. We choose a piece-wise linear activation function $\mathcal{F}_\beta(\cdot)$ as an activation function for $\beta(s)$, lower-truncating the input to zero and upper-truncating to β_{max} in order to avoid letting $\beta(s)$ take values greater than a satisfactory limit for which the discrete action decision will be made in an exploitative manner anyway, thus

$$\beta(s) = \mathcal{F}_\beta(\mathbf{b}^T \boldsymbol{\phi}(s)) \tag{12}$$

The covariance matrix $\boldsymbol{\Sigma}^a_s$ will be the diagonal $\sigma^2(s,a) \boldsymbol{I}_{m_a}$, such that $\sigma^2(s,a)$ will be the common variance of the Gaussians used as probability density functions for choosing the parameter values. A weight vector $\mathbf{s}_a$ is then assigned for the approximation of each $\sigma(s,a)$, such that

$$\sigma(s,a) = \mathcal{F}_\sigma(\mathbf{s}_a^T \boldsymbol{\phi}(s)) \tag{13}$$

where $\mathcal{F}_\sigma(\cdot)$ is a sigmoid activation function, adjusted to have a lower limit at σ_{min} and an upper limit at σ_{max}. For the updates of vector $\mathbf{b}$ and vector $\mathbf{s}_a$ which relate to the performed action a, a different strategy is perform depending on the sign and the value of $\bar{\delta}_V$. If $\bar{\delta}_V \ge 0$ then

$$\mathbf{b} \leftarrow \mathbf{b} + \mu_\beta \bar{\delta}_V (\beta_{max} - \mathcal{F}_\beta(\mathbf{b}^T \boldsymbol{\phi}(s))) \boldsymbol{\phi}(s) \tag{14}$$

$$\mathbf{s}_a \leftarrow \mathbf{s}_a - \mu_\sigma \bar{\delta}_V \mathcal{G}(\mathbf{s}_a^T \boldsymbol{\phi}(s)) \boldsymbol{\phi}(s) \tag{15}$$

where μ_β and μ_σ are parameters of choice, and $\mathcal{G}(\cdot)$ is the default sigmoid function $\mathcal{G}(x) = 1/(1+e^{-x})$. Note that $\bar{\delta}_V$ is

being used here, instead of δ_V used in the updates of $\boldsymbol{G}_a$. When $\bar{\delta}_V$ is negative a threshold value $\bar{\delta}_{thr} < 0$ is being used and the updates take place only if $\bar{\delta}_V \leq \bar{\delta}_{thr}$, such that

$$\mathbf{b} \leftarrow \mathbf{b} + \mu_\beta \bar{\delta}_V \mathcal{F}_\beta(\mathbf{b}^T \boldsymbol{\phi}(s)) \boldsymbol{\phi}(s) \tag{16}$$

$$\mathbf{s}_a \leftarrow \mathbf{s}_a - \mu_\sigma \bar{\delta}_V (1 - \mathcal{G}(\mathbf{s}_a^T \boldsymbol{\phi}(s))) \boldsymbol{\phi}(s) \tag{17}$$

The use of this threshold value is mainly needed on highly stochastic environments in order to avoid the positive feedback loop of exploration, which will be initiated from even small decrements of performance. Small decrements may not be the result of environmental changes but due to the stochastic nature of rewards. Not taking this into consideration will probably result in unnecessarily re-engaging exploration. The use of this threshold value can then be viewed as a changepoint detector which allows exploration increments only when significant reductions of performance occur.

Choose parameters $\alpha_{\{Q,C,V,A\}}, \mu_{\{\beta,\sigma\}}, \gamma, \beta_{max}, \bar{\delta}_{thr}$
Initialize $\boldsymbol{G}^a, \boldsymbol{W}, \boldsymbol{v}, \bar{\boldsymbol{v}}, \mathbf{b}, \mathbf{s}_a$
Observe the initial state s
while true **do**
  Estimate $\beta(s)$, $\sigma(s, a)$, $Q(s, a)$ with Eq.12,1,13
  Select the action tuple $(a, \boldsymbol{\theta}^a)$ with Eq. 10, 11
  Observe the new state s' and reward r
  Compute the errors $\delta_Q, \delta_V, \boldsymbol{e}$ with Eq.2,4,8
  Update the weights $\boldsymbol{W}, \boldsymbol{v}, \bar{\boldsymbol{v}}$ with Eq.3,5,6
  Compute the error $\bar{\delta}_V$ with Eq.7
  if $\delta_V > 0$ **then**
    Update the weights $\boldsymbol{G}^a$ with Eq.9
  end if
  if $\bar{\delta}_V > 0$ **then**
    Update the weights $\mathbf{b}, \mathbf{s}_a$ with Eq.14,15
  else if $\bar{\delta}_V < \bar{\delta}_{thr}$ **then**
    Update the weights $\mathbf{b}, \mathbf{s}_a$ with Eq.16,17
  end if
  Set $s \leftarrow s'$
end while

Algorithm 1: State Specific Parameterized Exploration.

Another approach of interest, involves cases where a reward signal is always present after each interaction with the environment. This is mainly the case in Human-Robot Interaction applications, where a global goal is present but also local myopic rewards should be taken into consideration. In such cases, the signal for adaptation $\bar{\delta}_V$ may be computed with the myopic mid and the long-term rewards gained from the present state. Specifically, we calculate $\bar{\delta}_V$ as

$$\bar{\delta}_V = V_m(s) - \bar{V}_m(s) \tag{18}$$

where $V_m(s)$ is the myopic value function (see supplementary material for a detailed explanation). Additionally, for tuning each action's uncertainty $\sigma(s, a)$, the difference $\bar{\delta}_Q$ of the mid- and long-term myopic action values $Q_m(s, a)$ may be used such that

$$\bar{\delta}_Q = Q_m(s, a) - \bar{Q}_m(s, a) \tag{19}$$

For updating the vector $\mathbf{b}$, the value of $\bar{\delta}_V$ is then used in Eqs.(14, 16), while the updates of vectors $\mathbf{s}_a$ are done by using the value $\bar{\delta}_Q$ instead of $\bar{\delta}_V$ in Eqs.(15, 17).

In general, the parameter search can be done in a normalized space and then be translated to the natural values (*e.g.*, for the actuators of the robot) with the use of an affine as $\boldsymbol{A\theta} + \boldsymbol{b}$ taking into consideration the range of each parameter as well as the minimum and maximum possible immediate rewards. With this consideration in mind, the full algorithm can now be summarized in Algorithm 1.

3 AI-based simulations

Although our goal is mainly to demonstrate the algorithm's adaptive nature in Human-Robot Interaction scenarios, we initially tested the algorithm on an environment which is more relevant to classic Artificial Intelligence (AI) / Machine Learning (ML) problems. On the other hand, this can also be viewed as a more general approach. As an example, a path planning problem is a generalization of many robotic tasks using the grid world as the discretized configuration space. However, since this is a new field of research, there is a lack of presence of benchmarks for parameterized action spaces. Here we present a modification of a grid world to an equivalent of a continuous maze as described in the next section.

For testing the performance of the algorithm and its adaptive nature, we consider a continuous maze as seen in Figure 2, which we name "matchball game". The objective is to "kick" the red ball, such that it does not fall off the surface of the maze and it finally reaches and hits the blue ball (target). If after an attempt the ball falls off the maze, it is repositioned at its last position prior to this attempt. Additionally the environment may change, by either removing a path and/or by adding new ones, by changing the target ball's position, or by changing any other environmental characteristics such as the floor's friction coefficient.

We consider an artificial agent that can kick the red ball in the 4 basic directions, so the discrete action space

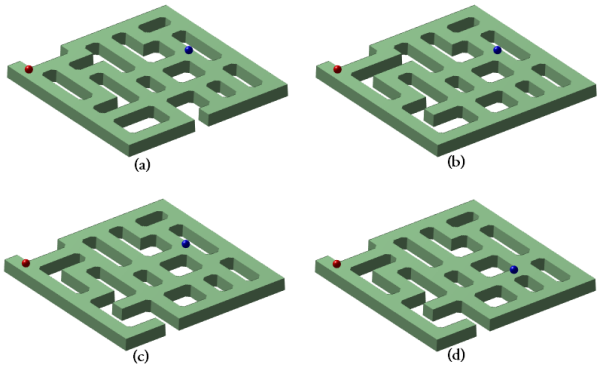

Figure 2: Experiment 1 - Different maze configurations used in the "matchball game" scenario. The red ball can be kicked in the 4 basic directions (up, down left, right) and the objective is to make it hit the blue ball without falling off the maze. Since a reward can be obtained only when the two balls hit each other, an optimal policy is one that minimizes the number of decision step, *i.e.*, kicking the red ball with the appropriate force at each step to make a single move within each corridor despite the floor's friction coefficient. The maze is continuous and the balls can be anywhere on its surface. If the red ball falls off the maze during an attempt, it is then reset at its last position on the surface prior to this attempt. Different maze configurations are used after each changepoint: **(a)** Initial maze (games 1-5000). **(b)** Maze used after the first changepoint (games 5001-10000). The optimal path is blocked and another path is opened. **(c)** Maze used after the second changepoint (games 10001-15000). The optimal path is blocked and another path is opened. **(d)** Maze used after the third changepoint (games 15001-20000). The position of the blue ball has changed.

is $A_d = \{up, down, left, right\}$. For each action then there is a continuous parameter $\theta^a \in (-3, 3)$ which is indirectly related to the initial velocity v_0 at which the ball will start the motion. Specifically, we consider an upper limit of v_{max} for the magnitude of the initial velocity, which practically corresponds to the maximum power with which the agent can kick the ball. Here, we perform the parameter search for θ values in the interval $(-3, 3)$, and the value is then transformed to the corresponding initial speed as $v_0 = (\theta + 3)v_{max}$. We find this approach convenient, since we will be using Gaussian exploration in a normalized parameter space. We also use an upper bound of $\sigma_{max} = 3$, which will result in almost uniform sampling of the parameter space in highly uncertain conditions. The ball's speed is reduced in time based on the environment's characteristics, and here due to simulation reasons we simply choose an update of $v_{t+1} = (1 - f_c)v_t$, where $f_c = 0.05$ is a friction-related parameter, while the ball stops completely if $|v_t| < \epsilon$, with ϵ being a small positive constant (a value of $\epsilon = 0.1$ was used). The state representation can be done by a number of ways as discussed in [29]. At first, we used radial basis functions, each one centered at a point $\mathbf{c}_i \in \mathbb{R}^2$ of a $N \times M$ grid. Then, if $\mathbf{x} \in \mathbb{R}^2$ is the 2D position of the ball on the surface, $\mathbf{d} \in \mathbb{R}^{N \times M}$ is a vector such that $d_i = \exp(-\|\mathbf{x} - \mathbf{c}_i\|^2 / 2\sigma^2)$, with σ being a parameter of choice. Using a continuous state vector such that $\boldsymbol{\phi}(s) = \mathbf{d}$ can then be one approach which however did not produce better results than using a one-hot representation vector, such that $\phi_i(s) = \mathbb{I}\{d_i = \|\mathbf{d}\|_\infty\}$, with $\mathbb{I}\{\cdot\}$ being the indicator function. A coarse coding strategy was also tried but due to the nature of the environment it only added computational complexity without further improving the performance.

Initially, the game begins as shown in Figure 2a. The agent is rewarded with a reward $r = 1$ if it eventually achieves to hit the blue ball and receives a zero reward in any other attempt. When a game is successfully completed the two balls are reset at their default positions, the number of steps taken is stored for evaluating the performance (take note that the minimum number of actions needed for an optimal policy is 3 in all the maze configurations used here) and a new game begins. In cases where a large number of actions have been made without hitting the blue ball (here this maximum was set to 1000), the game ends and the red ball is reset. After 5000 games have been completed, an environment changepoint occurs where the optimal path is blocked and another path opens such that the number of optimal steps remains the same as shown in Figure 2b. The agent is unaware of this changepoint occurrence and the objective would be to adapt and find the new optimal solution. At game 10000 another changepoint occurs by blocking the optimal path and creating a new one as seen in Figure 2c while at game 15000 the blue ball position changes as seen in Figure 2d. At game 20000 the simulations end and we regard this whole procedure as a hypersession.

3.1 Hyperparameter tuning

One of the algorithm's disadvantages is the large number of hyperparameters to tune. Using rewards in [0,1] the maximum value of inverse temperature β_{max} and the threshold value $\bar{\delta}_{thr}$ may be chosen manually, but an analytic derivation would be needed and it is out of the scope of the current work. Here we also included them in the hyperparameter search strategy and we only fixed the reward discount factor to $\gamma = 0.9$. Since we used the one-hot representation for the feature vector $\boldsymbol{\phi}(s)$, the parameter σ of the radial basis functions was irrelevant (*i.e.*, practically for any value of $\sigma \neq 0$ the representation vector will be the same for a given position $\mathbf{x}$ of the ball). Here we used $\sigma = 0.7$ and $N = M = 14$, resulting in a 196-dimensional state vector.

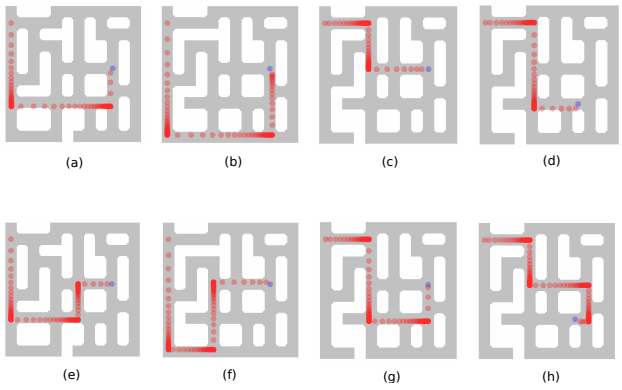

Figure 3: Experiment 1 - Paths. (a) A game instant of maze 1, where the optimal action-parameter tuples have been learned. In this game instant the red ball went very close to the edge of the surface. **(b)**Optimal path learned for maze 2. **(c)** Optimal path learned for maze 3. **(d)** Optimal path learned for maze 4. **(e)** A sub-optimal path learned for maze 1. **(f)** A sub-optimal path learned for maze 2. **(g)** A sub-optimal path learned for maze 3. **(h)** A sub-optimal path learned for maze 4.

Initially, we handcrafted all parameter values by observation in order to find a satisfactory set for initialization. We evaluated the performance by estimating the "optimality percentage" defined as the number of games that the agent completed at the minimum required actions over the total number of games in the hyper-session. Figure 3 shows the optimal paths (a,b,c,d) and some of the sub-optimal paths (e,f,g,h) found by the agents for each of the 4 mazes of Figure 2 respectively. Due to the rough representation of the state and the stochastic nature of the parameter value selection (we used $\sigma_{min} = 0.05$), each game instance is different and unique even when the algorithm's strategy has been stabilized. For example in Figure 3a, the optimal action-parameter tuples have been found, yet the red ball sometimes almost fell off the surface of the maze, as using σ_{min} is equivalent to adding an observable Gaussian noise to a deterministically chosen parameter value. There were also cases where the agent performed two or more sequential actions of the same type with small parameter values (*e.g.* two small steps) rather than one action with a larger parameter value (*e.g.* one big step), as also cases of "back-and-forth" movements.

We performed a uniform random search in a gradually decreasing hyper-cubic area around the dynamically changing optimal parameter set. The agent may also find a sub optimal path resulting in a lower optimality percentage. We ran the parameter search for 1000 hyper-sessions (that is 2×10^7 games in total) and the results for each trial can be seen in Figure 4a (the optimality-percentage for cases where a sub-optimal path was found is not shown). The parameters corresponding to the hyper-session with

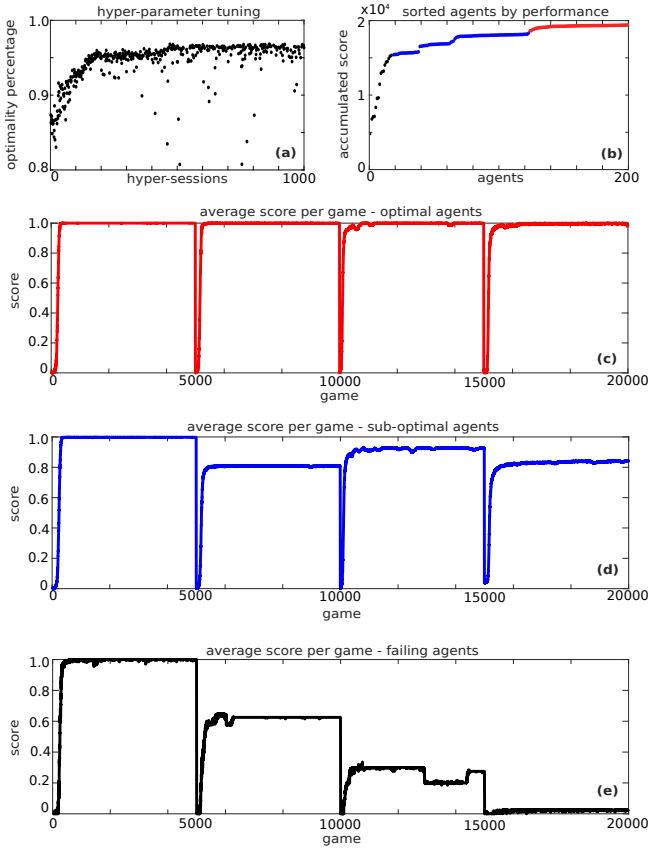

Figure 4: Experiment 1 - Parameter Tuning. (a) Hyper-parameter tuning strategy where each dot represents the performance of an agent for a chosen set of parameters. **(b)** Performance of 200 agents using the optimal parameter set. The red dots correspond to "optimal" type of agents, which adapted and found the optimal policy. The blue dots correspond to "sub-optimal" agents which adapted but didn't find the optimal policy. The black dots correspond to "failing" agents which adapted only for some changepoints but finally didn't find a successful policy up to the end of the hyper-session . **(c)** Average score per game gained by the "optimal" agents (128-200). **(d)** Average score per game gained by the "sub-optimal" agents. **(e)** Average score per game for the "failing" agents.

the larger optimality percentage are $\alpha_Q = 0.142$, $\alpha_A = 0.433$, $\alpha_C = 0.196$, $\alpha_V = 0.005$, $\mu_\beta = 0.205$, $\mu_\sigma = 1.378$, $\beta_{max} = 16.2$, $\bar{\delta}_{thr} = -0.311$, and they were chosen to further evaluate the performance of the algorithm as described in the next section.

3.2 Results

We ran 200 hyper-sessions for the chosen parameter set, and we tracked the number of actions performed by the agent to finish each game, regarding that an independent agent corresponds to each hyper-session. We sorted the agents with respect to their performance based on the fol-

lowing reasoning. If an agent ended a game in the optimal number of actions, a score of 1 was added to its total score. This value was reduced by 0.25 with each extra step taken, and we thus considered a null score for cases where the agent reached the blue ball in more than 4 extra steps. The accumulated score for each agent can be viewed in Figure 4b. About 39% of the agents (128-200) not only adapted to the changepoints but also learned to solve each game in the optimal number of steps. 56% (agents 11-127) adapted but found a sub-optimal path with on average only one extra action required for some of the game configurations (thus reaching the blue ball in 4 steps rather than the 3 steps required by the optimal policy). The remaining 5% (agents 1-10) adapted less and less well changepoint after changepoint. Figure 4(c,d,e) depicts the average accumulated score achieved by these three types of agents (noted with red, blue, black respectively). Overall, these results demonstrate the adaptive nature of the proposed algorithm in a quite complex task involving a continuous maze navigation with non-stationarities.

4 HRI-based simulations

As our main concerns and applications deal with human-robot interactions where the goal would be to maximize human engagement during an interaction task, for all the following experiments and setups we will be using a virtual engagement model as used in [16]. Previous researches on human-robot interaction have shown that the human engagement can be a critical aspect of the quality of the interaction [10]. Nevertheless, during interaction tasks the actions performed by a robot can have delayed effects on the human's behavior and on his engagement.

To mimic this, we chose a reward component to be given by a dynamical system which is based on the virtual engagement E of the human in the task. This engagement represents the attention that the human pays to the robot and will constitute a reward signal, since this type of joint attention social signals have been shown to activate the same brain regions that are activated by non-social extrinsic rewards such as food or money [30]. In all the simulations followed, the quantified engagement starts at 5, increases up to a maximum $E_M = 10$ when the robot performs the appropriate actions with the appropriate parameters, and decreases down to a minimum $E_m = 0$ otherwise:

$$E_{t+1} = \begin{cases} E_t + \eta_1(E_M - E_t)H(\boldsymbol{\theta}_t^a), & \text{if } a_t = a^\star \,\&\, H(\boldsymbol{\theta}_t^a) \geq 0 \\ E_t - \eta_2(E_m - E_t)H(\boldsymbol{\theta}_t^a), & \text{if } a_t = a^\star \,\&\, H(\boldsymbol{\theta}_t^a) < 0 \\ E_t + \eta_2(E_m - E_t), & \text{otherwise} \end{cases} \quad (20)$$

where $\eta_1 = 0.1$ is the increasing rate, $\eta_2 = 0.05$ is the decreasing rate, and $H(\boldsymbol{x})$ is the re-engagement function given by

$$H(\boldsymbol{x}) = 2\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}^\star)^T(\boldsymbol{\Sigma}^\star)^{-1}(\boldsymbol{x} - \boldsymbol{\mu}^\star)\right) - 1 \quad (21)$$

where $a^\star$, is the optimal discrete action, $\boldsymbol{\mu}^\star$ is the optimal parameter vector $\boldsymbol{\theta}_\star^a$ for the optimal action and $\boldsymbol{\Sigma}^\star$ is a diagonal matrix $\sigma^{\star 2}\boldsymbol{I}$ of size $m_{a^\star} \times m_{a^\star}$. To picture the idea, the parameters for which $H(\boldsymbol{x}) = 0$ define the boundaries of an $m_a^\star$-dimensional ball in the parameter space, inside which the engagement is increased. In general, each parameter might have difference tolerance, however for all the experiments we will be using a common $\sigma^\star = 10$, while all parameter values will be in [-100,100]. Figure 5 depicts H-function in the case where the optimal action has only one continuous parameter.

In the cases for which the application has a dual learning objective of both completing a task and maximizing engagement, a hybrid reward shaping with components r^e and r^c will be used, where the component r^e is relevant to the maximization of the engagement will be given by:

$$r_{t+1}^e = E_{t+1} + \lambda \Delta E_{t+1} \quad (22)$$

where $\lambda = 0.7$ is a weight. This will ensure that the algorithm gets rewarded in cases where the engagement is low but nevertheless has just been increased by the action tuple $(a, \boldsymbol{\theta}^a)$ performed by the robot. The component r^c will depend on the task. In case the objective is only to maximize the engagement (which here is our main goal), only r^e will be used as a feedback to the agent. For the HRI experiments, we use the signals $\bar{\delta}_V$ and $\bar{\delta}_Q$ to tune the inverse temperature and the action-specific uncertainty of each state respectively, as described in section 2.

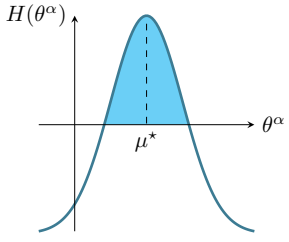

Figure 5: Principle adopted to simulate variations of child engagement as a function of the distance between the robot's current continuous parameters of action and optimal ones.

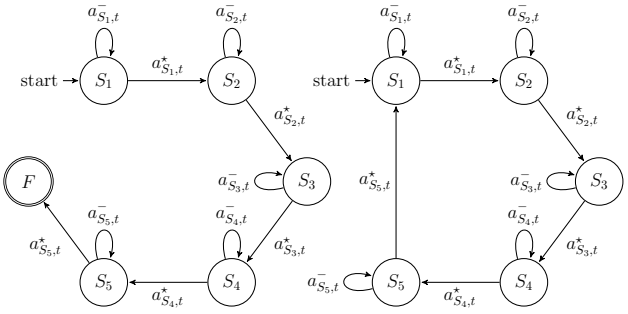

Figure 6: Experiment 2. Left: MDP for a single session. The optimal action $a^{\star}_{S_i,t}$ on each state S_i results in a state change while all other actions $a^{-}_{S_i,t}$ are sub-optimal. The transitions are not independent on the parameter value. **Right :** An equivalent MDP used for repeatedly simulating numerous sessions.

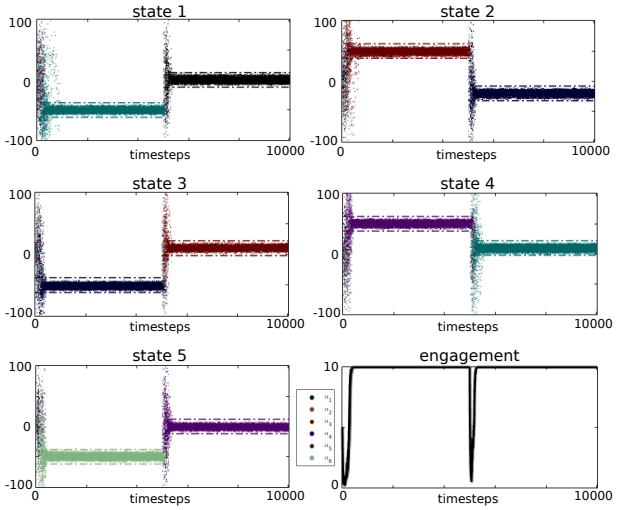

Figure 7: Experiment 2 Task 1. The chosen action-parameter tuples shown as colored dots for 50 hyper-sessions. The color of the horizontal dash-dot lines represent the optimal actions while the range between them represents the tolerance interval. The engagement on each timestep is shown at the bottom right.

4.1 Experiment 2

As an extension of previous experiments presented in [15], here we test our algorithm on a parameterized action Markov Decision Process of 5 states plus a final accepting state (left MDP of Figure 6) where the objective is to maximize the virtual engagement function. The action space is $A = A_d \times A_p$, where $A_d = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $A_p = [-100, 100]$; simply there are 6 discrete actions, each one with a continuous parameter. We assume that for every state s_i at each timestep t, there is an optimal action $a^{\star}_{s_i}$ which leads to the next state in a deterministic manner (*i.e.*, $P(s_{i+1}|s_i, a^{\star}_{s_i}) = 1$) independently of the parameter choice $\theta^{\alpha}_{s_i}$, while all other actions $a \in A_d \backslash \{a^{\star}_{s_i}\}$ result in no state change. However each optimal discrete action $\alpha^{\star}_{s_i}$ is characterized by an optimal value $\mu^{\star}_{s_i}$ as shown in Figure 5, and choosing a parameter $\theta^{\alpha}_{s_i}$ such that $H(\theta^{\alpha}_{s_i}) \geq 0$ will result in an increase of the reward signal r_t as described in Eq. 22. This occurs when:

$$\mu^{\star}_{s_i} - \sigma^{\star}\sqrt{2ln2} \leq \theta^{\alpha}_{s_i} \leq \mu^{\star}_{s_i} + \sigma^{\star}\sqrt{2ln2}$$

where $\sigma^{\star}$ will be a global parameter for all states and actions. This inequality specifies a tolerance interval with a fixed width, but with non-fixed boundaries since the optimal action-parameter tuples are non-stationary. Having a reward feedback after each action on each visited state don't truly demonstrate the learning ability of an RL agent, and the problem could be tackled by having independent multi-armed bandit agents to each state. However, the propagation of rewards from future states wouldn't take any effect (which here does), and the optimal strategy for multi-objective cases would be untraceable.

At first we perform numerical simulations and measure the performance on a dynamic task with a global changepoint occurrence of the optimal action-parameter tuples. We then demonstrate the main adaptive nature of the algorithm on an environment where local state changes occur, both drifting and abrupt. We choose the parameters by at first manually tuning an initial set based on observations, and then using the method presented in section 3.

4.1.1 Global changepoints

We simulate the algorithm on a task (*Task 1*) with a global changepoint of the optimal action-parameter tuples, using the right MDP of Figure 6 for 10000 timesteps (we call this a hyper-session). For $1 \leq t < 5000$ the optimal actions on each state are $\{a^{\star}_{s_1,t}, a^{\star}_{s_2,t}, a^{\star}_{s_3,t}, a^{\star}_{s_4,t}, a^{\star}_{s_5,t}\} = \{a_2, a_3, a_4, a_5, a_6\}$ and their corresponding optimal parameter values are $\{\mu^{\star}_{s_1,t}, \mu^{\star}_{s_2,t}, \mu^{\star}_{s_3,t}, \mu^{\star}_{s_4,t}, \mu^{\star}_{s_5,t}\} = \{-50, 50, -50, 50, -50\}$. For $t \geq 5000$ the optimal discrete actions change to $\{a^{\star}_{s_1,t}, a^{\star}_{s_2,t}, a^{\star}_{s_3,t}, a^{\star}_{s_4,t}, a^{\star}_{s_5,t}\} = \{a_1, a_4, a_3, a_2, a_5\}$ and their parameters to $\{\mu^{\star}_{s_1,t}, \mu^{\star}_{s_2,t}, \mu^{\star}_{s_3,t}, \mu^{\star}_{s_4,t}, \mu^{\star}_{s_5,t}\} = \{0, -10, 10, 10, 0\}$. Figure 7 captures the results of *task 1* after 50 hyper-sessions. The graphs depict the chosen action-parameter pairs with colored dots for each timestep that the agent was found in one of the 5 states. The color of the horizontal dash-dot lines represents the optimal action and the range between these lines is the tolerance interval (*i.e.*, the values of the parameters for which the

engagement is increased). The graph at the bottom-right shows the virtual engagement.

The algorithm performs exploration at first (shown by the large "cloud" of actions) and then manages to approximate the optimal action-parameter tuple $(a^{\star}_{s_i,t}, \mu^{\star}_{s_i,t})$ for all states s_i (as the colored dots fall inside the tolerance interval). The uncertainty of the chosen parameters, which is adaptively tuned with the use of the sigmoid $\mathcal{F}_{\sigma}(\cdot)$, described in the mathematical section, has been lower bounded with a value of $\sigma_{min} = 2$, which is here pictured by the minimum width of the action "cloud". The engagement at first drops but then increases to a value of 10, achieving perfect performance right before the changepoint occurrence. Right after the changepoint, both exploration and action uncertainty increased resulting in finding the new optimal actions and achieving a perfect performance after 200 timesteps on average.

Interestingly, Suppl. Figure S1 and S2 in the supporting material show that the non-state specific exploration of the previous version of the algorithm [16] – nevertheless extended to deal with multiple states – can learn this type of tasks with a performance below the one of the new algorithm. In Suppl. Figure S1 the global changepoint only requires adaptation of the continuous parameters for the stable optimal actions. In this case, the engagement drops mildly after the changepoint (resulting in a partial re-exploration) and then re-converges to a good but non-optimal engagement (slightly below 9). In Suppl. Figure S2 the global changepoint require adaptation of both the discrete actions and the continuous parameters, as in Figure 7, which results in a sharp drop of engagement (and thus a full re-exploration). In this case, the previous version of the algorithm enables to re-converge to approximately the same level of engagement than before the changepoint. This highest engagement is nevertheless below the one with the novel algorithm. Nevertheless, the main message here is that a global changepoint results in sufficient cumulated novelty over all states to permit a satisfying performance with global active exploration. In the next section, we will show that local changepoints result in much more subtle variations of uncertainty on action parameters so that a state-specific active exploration is required for optimal performance.

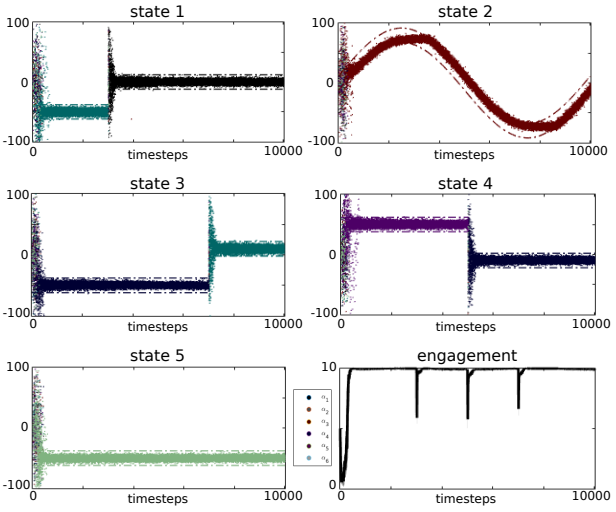

Figure 8: Experiment 2 Task 2. As in Figure 7, the chosen action-parameter tuples for 50 hyper-sessions are shown as colored dots. The tolerance interval is represented by the range between the dash-dot lines and the type of the optimal discrete action is represented by their color. The engagement is shown at the bottom right.

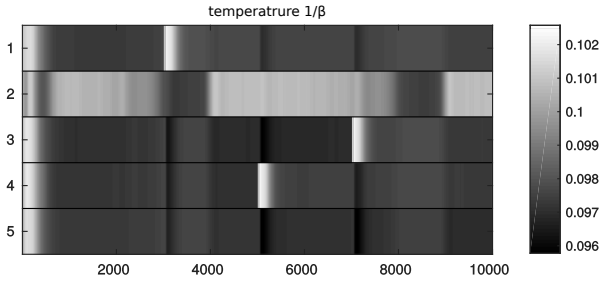

Figure 9: Experiment 2 Task 2. Average temperature levels ($T_t(s_i) = 1/\beta_t(s_i)$) for all states s_i from 50 hyper-sessions of *task 2*. Each row i of the image corresponds to a state s_i and each column corresponds to a timestep t. Bright regions are relevant with high exploration levels while dark regions correspond to more exploitative strategies.

4.1.2 Local changepoints

Here we test the algorithm on *Task 2*, for which states follow different dynamics as shown in Figure 8. In state s_1 a changepoint occurs at timestep $t = 3000$. The optimal action-parameter tuple for $t < 3000$ is $\{a_2, -50\}$ and changes to $\{a_1, 0\}$ for $t \geq 3000$. In state s_2 the optimal discrete action is a_3, however the optimal parameter is changing sinusoidally in time. In state s_3 a changepoint occurs at timestep $t = 7000$, where the optimal action-parameter tuple is $\{a_4, -50\}$ for $t < 7000$ and $\{a_2, 0\}$ afterwards. State s_4 is also subject to an abrupt change, where the optimal tuple is $\{a_5, 50\}$ for $t < 5000$ and $\{a_4, -10\}$ for $t \geq 5000$. State s_5 is stationary, with $\{a_6, -50\}$ as an optimal action tuple.

From the results shown in Figure 8, we can see that the optimal actions and their associated optimal parameters are approximated in an accurate manner. The action "cloud" falls inside the tolerance interval most of the

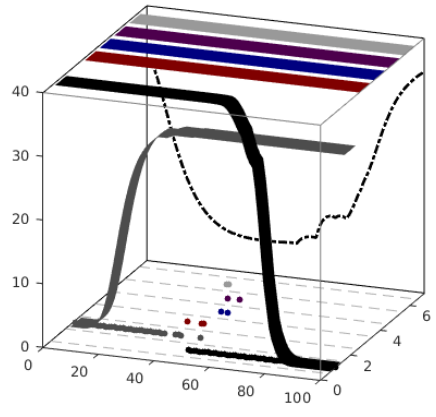

Figure 10: Experiment 2 Task 2 (stochastic version). A block describing the sequence of selections of the 6 discrete actions (different colors, y-axis) together with the exploration parameters $\beta(s)$ and $\sigma(s, a)$ (z-axis) in state s_1 during a random session right after a changepoint occurrence. Time starts from the changepoint and shows the first 100 timesteps (x-axis) where the agent was in state s_1. Action 2 was optimal before the changepoint, and Action 1 after. The height of each ribbon i illustrates the value of $\sigma(s_1, a_i)$ (the closer to 0 the more exploitative). The colored dots at the bottom correspond to the discrete actions taken, each one shown on a different line for clarity. The dashed dotted line on the back of the cube corresponds to the inverse temperature $\beta(s_1)$ (the closer to 0 the more exploratory).

time, and the engagement is close to 10 even though the optimal parameter value of action a_3 in state s_1 is constantly changing, and the exploration results in tracking the dynamics of the environmental changes. It is also important to note that the exploration in state s_5 does not increase, even at the moments where the rest of the environment is subject to abrupt changes. Figure 9 shows the average temperature levels $T(s) = 1/\beta(s)$ in all states and all timesteps. In state s_1, the reduced immediate rewards $r_t(s_1)$ result in a decrease of the inverse temperature $\beta_t(s_1)$ which is shown by the increase of intensity in the first row of the image. In state s_2 where the optimal parameter values are drifting, the temperature stays high except for the small time intervals where the monotonicity of the sinusoid changes and the existing approximation falls inside the tolerance region.

To have a better picture of the active exploration strategy taken, Figure 10 shows an example for a non-optimized version of the algorithm with stochastic rewards (the parameters used here are those that had been optimized for a different task in [16]). After the changepoint in state s_1 the uncertainty of parameter values starts to increase, while the inverse temperature decreases. After uncertainty and exploration reach high levels, the agent starts to choose among other discrete actions (shown as color dots at the bottom of the figure). When the optimal action is found and the chosen parameter starts to fall inside the tolerance interval, the performance starts to re-increase resulting in a reduction of the uncertainty of this new action and re-exploitation phase.

Importantly, when comparing the performance in Expertiment 2 Task 2 (local changepoints) with that of the previous non-state specific exploration (Suppl. Figure S4 in the supporting material), we found that the new algorithm outperforms the old one. Strikingly, Suppl. Figure S4 shows that each local changepoint in a given state pollutes performance in the other states (timesteps 3000, 5000 and 7000). More importantly, the non-state specific exploration version of the algorithm completely fails in capturing the drifting type of environmental change. Overall, the old algorithm still manages to fastly adapt to each abrupt changepoint and the simulated engagement is most of the time higher than 8/10 although not optimal. The main message here again is that state-specific active exploration is required for optimal performance with the subtle variations of uncertainty in action parameters locally induced by local changepoints.

4.1.3 Robustness on stochasticity and volatility

To evaluate the algorithm's performance on stochastic environments and to further investigate its limits of adaptivity based on the volatility of the environment, we performed a number of experiments on stochastic version of Experiment 2 Task 1 (global changepoints) by altering the probability of a correct transition from 0.5 to 1. Specifically, if the chosen action was the optimal one, then the transition to the next state took place with probability p and the state did not change probability $1 - p$. Similarly, when a non-optimal action was chosen the state remained the same with probability p, while the state transitioned to the next one with probability $1 - p$. While we investigated the performance on different values of p from 0.5 to 1 with a step of 0.05, we also changed the volatility of the environment in 2 different ways, abruptly changing and drifting.

At first, we changed the optimal action-parameter tuple in a circular manner based on the following reasoning. If n is an incremental index (initialized at zero) such that $n \leftarrow n + 1$ after every changepoint, then the optimal action at each state s_i is given by $a^{\star}_{s_i} = a_j$, with $j = \mod(i + n - 1, 6) + 1$, while the optimal parameter value is $\mu^{\star}_{s_i} = (-1)^{n+i} \times 20$. With N_{cp} being the number of changepoints (uniformly distributed over a time horizon of 10000 timesteps), we altered its value from 0 to 10 and

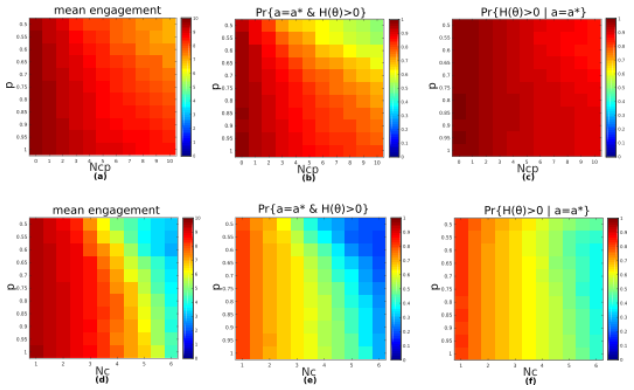

Figure 11: Experiment 2. (a,b,c) Performance for different stochasticity levels p (denoting the probability of a correct transition) and the number of changepoints N_{cp} in a globally changing environment. **(d,e,f)** Performance for different stochasticity levels and cycles N_c of sinusoidal changes of all parameter values in the specified time horizon of $T = 10000$.

we evaluated the performance of the algorithm after running 20 hyper-sessions for each pair of (p, N_{cp}). We computed the average engagement achieved and numerically estimated the probabilities $Pr\{a = a^{\star}\&H(\theta^a) > 0\}$, and $Pr\{H(\theta^a) > 0|a = a^{\star}\}$ as shown in Figure 11(top).

Secondly, we fixed the optimal actions to some random set and we altered the parameter value sinusoidally in timesteps, as done in state 2 of *Task 2*. With N_c being the number of cycles of the sinusoid up to the end of a hyper-session, we tried values from $N_c = 1$ to $N_c = 6$ with a step of 0.5 and we evaluated the performance on all pairs (p, N_c). The results are shown in Figure 11(bottom).

To have a reference point, the performance of the global changepoint in Experiment 1 Task 1 (shown in Figure 7) corresponds to the mean engagement value of Figure 11a for $p = 1$ and $N_{cp} = 1$. Reducing p then seems to have had a low impact on the performance. However, the performance gradually decayed while N_{cp} increased and p was lowered. In the case of "drifting" type of environments, the mean engagement was high for 1-3 cycles but steeply reduced after this point.

4.2 Experiment 3

In the following experiment we performed visual simulations of a child-robot interaction in a Matlab environment, where 3 colored cubes were placed randomly at 3 positions (left, center, right). The objective was for the robot to learn the "Tower of Hanoi" game and build a tower of cubes at the center position by only having a visual feedback from the child to estimate her current engagement (social reward) and a small reward every time the task was completed (non-social reward). The robot's actions consisted in grasping and placing objects. It could also follow the game's rules by not placing a larger cube over a smaller one and had the ability of parameterizing the trajectories of its movements (here the only considered continuous parameter was be the speed of movement). We considered 27 states and used a one-hot representation as a feature vector. Specifically, $\boldsymbol{\phi}(s)$ was a 27-dimensional vector where all components were zero except the one corresponding to the current state s. There were 6 actions in total with action a_1 being to grasp the top cube at the "left" position and place it at the "center". With using "LC" as abbreviation for this type of action, the discrete action space was then $\{a_1, a_2, a_3, a_4, a_5, a_6\} = \{LC, LR, CL, CR, RL, RC\}$ (however not all of the actions were "legal" and available in every state). The parameter values were varying in the range [−100, 100], as in the MDP case, and were then translated to the appropriate interval for the robot's actuators.

The child's preference on the parameter values was related to the difficulty of the action. To be more precise, we broke down the parameters to 6 families of actions, where each family had similar characteristics of demanding dexterity. We considered actions of *type 1* the actions for which a "lone" cube (cube that is not over another cube) was placed at a "lone" position (positions where there is no cube) and therefore the natural speed of performing this action was relatively fast (as there was no particular difficulty). We used the abbreviation "1 − 1" for such actions and they all shared a common optimal parameter value $\mu_1^{\star}$. Actions of *type 2* were the actions for which the top cube from a "stack of two" was picked and placed at a "lone" position. We used the abbreviation "2 − 1" for this type of actions and they all shared the same optimal parameter value of $\mu_2^{\star}$. Using the same reasoning, there were 6 different types of actions {"1 − 1", "2 − 1", "3 − 1", "1 − 2", "2 − 2", "1 − 3"} in an ascending order of dexterity requirements, and we may assume that the child's preference (*i.e.*, here corresponding to her possible expectation for natural movements) was correlated with actions that display a more "natural" behavior. However, while the task was being learned, the simulated child could become more and more impatient, resulting in an iterative increment of the optimal parameter values. This way we forced the robot to display adaptive characteristics of learning in such parameterized-action environments. Note that the algorithm could learn arbitrary speeds for each (state, action) pairs, and arbitrary variations of optimal speed (*e.g.*, decrements of speed). But here, the requirements of the task were chosen to illustrate a more meaningful child-robot interaction.

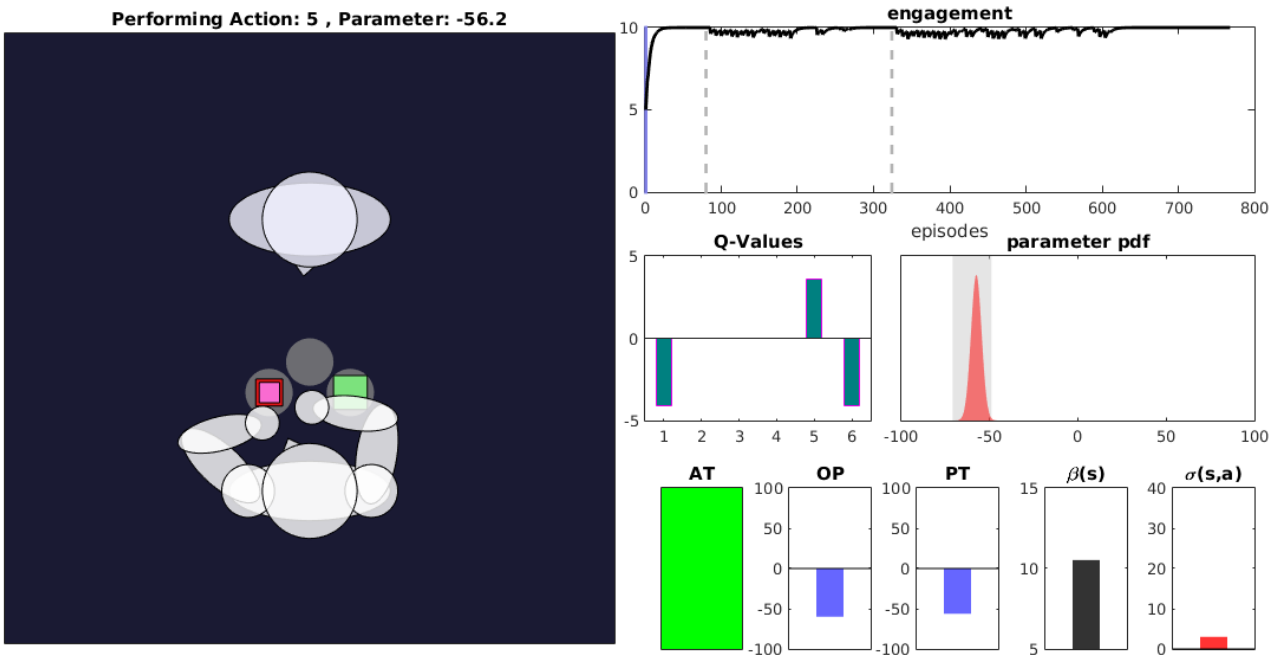

Figure 12: Experiment 3 Tower of Hanoi Game. Left: an instance of the simulation where the robot has just placed the small-sized pink cube over the medium-sized red cube. AT: Action Taken (green for optimal-red for bad action) OP: Optimal Parameter, PT: Parameter Taken, $\beta(s)$: inverse temperature in this state, $\sigma(s, a)$: standard deviation for the Gaussian exploration, Q-Values: The action values in this state for the "legal" actions. On the middle right the tolerance interval is shown with a shaded area and the distribution from which the parameter is shown with a red color. The top graph depicts the engagement achieved, where the dash dot lines show the time steps at which an optimal parameter value of an action changed.

Figure 12, shows an instance of the simulation environment after learning where the robot had already placed the small-sized pink colored cube (previously placed over the large-sized green cube) over the medium-sized red at the left position. The child's engagement is visually displayed, representing the relation of the angle of the child's gaze to the point of interest. The gaze angle was sampled from a bimodal distribution centered on the optimal angle (*i.e.*, towards the point of interest), and for which the distance of the modes were inversely proportional to the engagement values. The point of interest here was the position of the robot's moving arm performing an action, we could however generalize to have multiple points with different weights (this is out of the scope of the present work). At the bottom right, the green colored box shows that the action performed was the optimal one (otherwise the color would be red). The "OP" corresponds to the optimal parameter value for the action taken and the "PT" corresponds to the parameter tried. The inverse temperature, the uncertainty of action parameter and the Q-Values are shown. As the pink-colored cube was initially over the large green cube, the Q-value of optimal action a_5 was the highest among the allowed actions to take. The middle-right figure shows the tolerance interval with a shaded area, together with the probability density function from which the current parameter value was sampled. The top-right graph displays the engagement achieved, while the gray dashed dotted lines are placed at the timesteps on which an optimal parameter of an action increased. Suppl. Video 1 in the supporting information illustrates a simulation including different task phases: (1) first a simulation of the task with a pretrained robot that already knew how to solve the Tower of Hanoi task but had nevertheless to adapt its continuous parameter of action (*i.e.*, speed of movement) to the child; (2) then a sequence of changepoints where the optimal speed of movement was iteratively increased; (3) finally, a "new born" robot which had to relearn the task from scratch. Note the smooth variations in the speed of movement which was generated by the algorithm and which produces an apparent more "natural" behavior during the social interaction.

Finally, to investigate the adaptive limits of the algorithm for this specific environment, we evaluated the performance on a number of experiments where a changepoint of the optimal parameter occurred for all types of actions every T_{cp} number of timesteps, in a sequential manner. We altered T_{cp} to take values in $\{10, 50, 100, 500\}$ and we ran 20 hyper-sessions for each case. The achieved

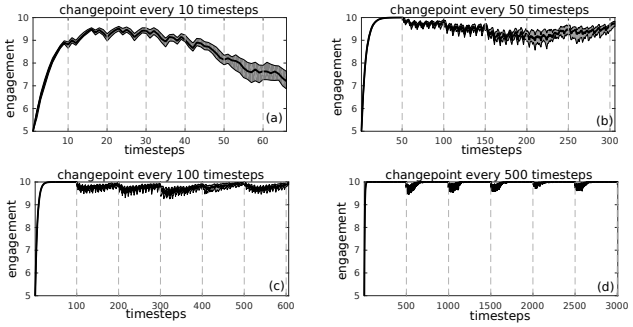

Figure 13: Experiment 3. Average engagement of a pre-trained agent for 20 hyper-sessions in locally changing environments, where a changepoint occurred **(a)** every 10 timesteps, **(b)** every 50 timesteps, **(c)** every 100 timesteps, **(d)** every 500 timesteps.

mean engagement can be viewed in Figure 13. For $T_{cp} = 10$ the algorithm could not catch the dynamics and the engagement dropped. For $T_{cp} = 50$, the algorithm managed to adapt partially. For $T_{cp} = 100$, the engagement maximized right before the next changepoint occurred, while for $T_{cp} = 500$, there was enough time for the engagement to be maximized and stabilized after each changepoint.

4.3 Experiment 4

In order to illustrate the different ways children engage in a collaborative game with a robot, we performed realistic simulations of the child-robot interaction pilot task described above (Section 2.1). We implemented the simulations in the virtual robot experimentation platform (V-REP). In the considered scenario, a small humanoid robot, in this case a NAO, interacts with a human child subject, and the goal is to collaboratively perform a task involving pointing at, picking up and placing objects (cubes) located in the scene (Figure 14) in order to later use them for the construction of a tower. Ideally, when the robot points at a cube, the child should understand the robot's intention, pick up the cube and hand it over to the robot so that the game continues. In this case the child is maximally engaged and the robot executes the pointing action with a specific child-dependent way that appears more natural and intuitive to the child. With this simulation, we do not attempt to visualize the whole tower-building game but to show how the robot' s expressive gestures lead to more engaging interactions with human partners.

In the pilot version of this joint attention experiments that we performed between a NAO and children (Section 2.1), we studied how the robot's movements affected the child's engagement. The goal was again the pick-and-place task described above and we observed that when the robot opened and closed its grip or exchanges glances between the child and the object for a period of time while pointing at the object, it contributed to an increase of the child's engagement. Nevertheless, we have yet too little data with this pilot and simulations appear crucial to fine tune the algorithm for this scenario.

Inspired by these experiments, we parameterized the simulated pointing action of the robot with low-level features describing the physical aspects of the robot's movements. These movements accompanied and reinforced the robotic pointing action by iteratively opening-closing its hand or alternating glances at the child and at the object. The amount of time of these iterations was an aspect of the gesture's expressivity and also a parameter that had to be learned. Depending on the parameters values, the robot executed a pointing gesture of different expressivity. Examples of such gestures are shown in Table 1. In the current implementation of our simulations, the robot performed one discrete action (pointing gesture) which had two parameters (t_1, t_2) corresponding to the time in seconds of the open-close and glance movements respectively.

Table 1: Robot's pointing action with parameters corresponding to increasing levels of expressivity.

	Pointing gesture
expressivity ↓	point ($t_1 = 0, t_2 = 0$)
	point + open-close hand ($t_1 \neq 0, t_2 = 0$)
	point + exchange glance ($t_1 = 0, t_2 \neq 0$)
	point + open-close + glance ($t_1 \neq 0, t_2 \neq 0$)

In the lowest level of expressivity, the robot only performed a pointing gesture by simply putting its arm forward towards the object. In the next levels, the pointing actions were reinforced by iteratively opening-closing the hand or exchanging glance between the child and the object for t_1 and t_2 seconds respectively. We assumed that each simulated child reached an optimal state of engagement when the pointing action was performed by the robot with specific child-dependent durations of open-closing and glancing. For instance, one of the children reacted and came to help the robot only when the robot gazed at her, while another child did not like to be looked at by the robot. Therefore, the robot had to be able to adapt to different children and to learn their corresponding parameters. Nevertheless, we also considered that the algorithm could be initialized based on the parameters obtained on average during previous interactions with children. This way,

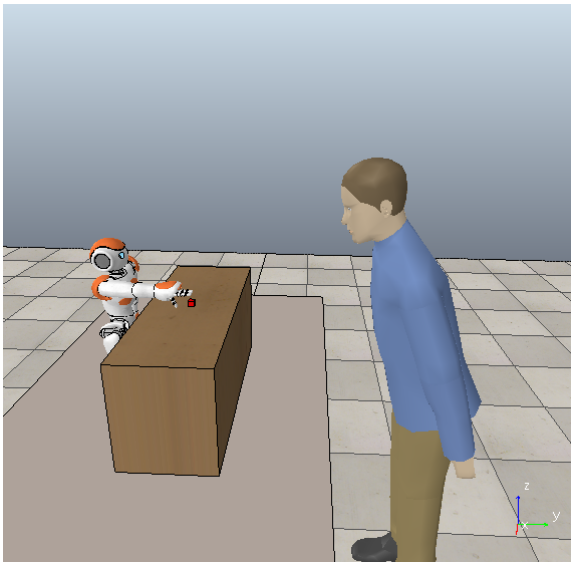

Figure 14: Experiment 4. Setup used for the simulations of the same task as the pilot real child-robot interaction experiment.

the algorithm started from a meaningful average value of action parameters/durations $(\bar{t_1}, \bar{t_2})$, rather than being initialized randomly, and then adapted to each specific child. In future work, we can study whether the average parameters over different interacting children is efficient or whether there exists distinct clusters of parameters – especially within the data obtained in the real experiments – that should be used as separate initialization points.

In our simulations, we defined a time range from 0 to 10 seconds. Figure 15 shows an example of the parameters adaptation from their default to their optimal values. Specifically, the robot firstly interacted with an "average child", meaning that the child engaged optimally with parameters $(\bar{t_1}, \bar{t_2})$. Then, at timestep 40, the experiment ended and another child (child 1) with different optimal parameters started interacting with the robot. As mentioned before, to accelerate learning the robot did not reset its action parameters but adapted to its new partner. The engagement of child 1 was initially very low but progressively increased as the robot was finding the optimal continuous action parameters. Similarly, at timestep 80 child 2 took the place of child 1 and the robots readjusted its parameters. After running this series of experiments 10 times and plotting the mean and standard deviation of the action parameters and the children's engagement (Figure 15), we observe that in less than 10 timesteps the robot found the optimal parameter values of child 1 whose engagement reached 8 in just a few timesteps. The adaptation was similar for child 2.

Suppl. Video 2 in the supporting information illustrates a simulation where the algorithm was successively interacting with three different simulated children. Each child had its own preferences in terms of expressivity of action by the robot and progressively got more and more engaged as the robot was reaching the appropriate level of expressivity. Child engagement was here simulated from looking away (level 0) to looking from far away (level 1), getting closer (level 2), bending the torso (level 3) as the robot was progressively improving its behavior.

5 Discussion and conclusion

In this work, we proposed a novel state-specific active exploration process in parameterized action spaces (*i.e.*, combining the learning of discrete (state,action) values and a vector of continuous action parameters associated to each (state,action) pair). This algorithm permits to benefit from the task decomposition into a small set of discrete actions, that can be more easily categorized and understood by children during child-robot interaction, and at the same time to finely tune in an online manner continuous parameters of actions that can be adjusted in a child-specific way (*e.g.*, duration of action, speed, gaze orientation, expres-

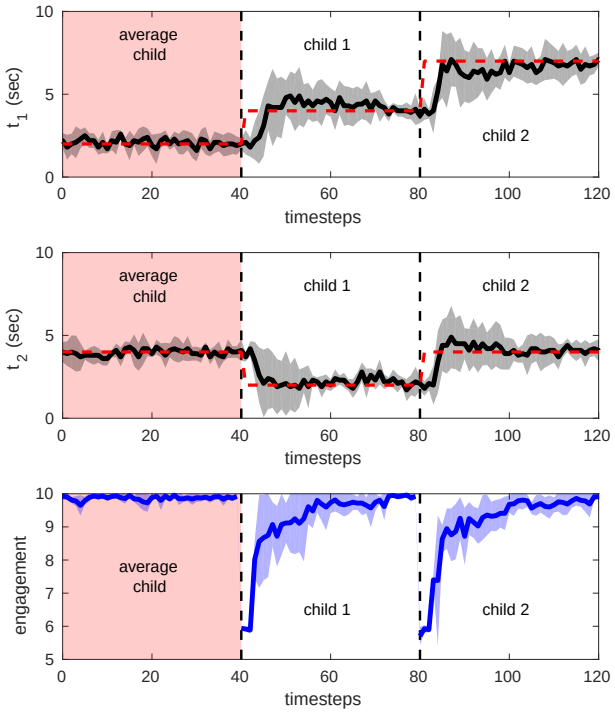

Figure 15: Experiment 4. Left: Before timestep 0 the robot executed the default parameters values, no adaptation was performed. After timestep 0, the robot adapted its action parameters (black) towards the optimal action parameters (red). **Right:** Child's engagement reached 90% within less than 10 trials.

sivity of action, etc.). This algorithm extends our previous non-state-specific active exploration proposed in [15, 16] which was restricted to single state scenarios (where each action immediately results in an outcome). The extension both permits to deal with multi-state scenarios (where an action's effects can be delayed in time) and to cope with local non-stationarities in the state space without interfering with learning in other stable portions of the state space (*e.g.*, if the child changes her expectation about what the robot shall do with a particular object, the robot can locally re-explore its behavior for this object while maintaining a stable behavior elsewhere). We presented a series of simulated child-robot interaction experiments, showing (1) that the new algorithm outperforms the previous version; (2) that it can perform a Tower of Hanoi experiment where in addition to achieving the task within the minimum number of steps the algorithm has also to learn to build the tower with the appropriate speed of movement for the child with whom it interacts; (3) and that it can perform a task where the robot needs to find the child-specific appropriate level of expressivity of action while pointing at an object in order to make the child engage into joint action.

This work has important implications for machine learning methods applied to robotics as well as more specifically for the field of human-robot social interaction. From a theoretical point of view, the proposed algorithm provides a novel way to perform state-specific active exploration in reinforcement learning, thus expanding active exploration methods [19–23]. Moreover, it expands such active exploration to parameterized action spaces [17, 18], thus enabling to dynamically adjust in parallel choices between discrete actions and continuous parameters of action. In terms of robot learning implications, the fast adaptation which is permitted by such a state-specific active exploration algorithm constitutes a promising intermediate solution between discrete and continuous action learning methods that have so far been applied to robotics (see [14] for a recent review). From the point of view of the possible applications to human-robot social interaction, the algorithm represents a new step on the so far little developed investigation of robot learning abilities to adapt online to variations of non-verbal social signals [1, 5–9]. More particularly, the algorithm represents a solution to make robots adapt to variations in human engagement [7, 9, 10]. Finally, for educational applications where a small humanoid robot can assist a human teacher, the proposed algorithm constitutes a novel tool that may contribute to promoting typical or Autistic Spectrum Disorders (ASD) children's interest in educative games, and help further develop their social skills [2–4] and enhance their motivation [5].

Two particular features of the algorithm deserve to be highlighted in terms of the potential they have for child-robot interaction. The first one is the general stability of the robot's learning performance during social interaction (at least proven here in simulation) thanks to the active exploration process. On the one hand, the fact that an improvement in robot performance leads to more exploitation enables a strong stability of behavior when the conditions are stable. This contrasts with learning algorithms with a fixed exploration rate which can remain suboptimal and unstable even during the stable task phases after learning. On the other hand, transient re-exploration and thus fast adaptation is still permitted at any time a changepoint is detected. This contrasts with learning methods where the exploration phase is circumscribed with prior information about the expected duration of learning. Thus here the active exploration process enables to alternate between transient periods of exploration and fast re-learning, and periods of stabilization during exploitation. The second important feature is the modularity of the combination of discrete and continuous action parameters, as well as associated exploration rates for each of them. This permits applications where all are updated in parallel, as well as applications where one module is frozen. For instance, one may want the robot to have a fixed pre-learned behavioral policy in the discrete domain (for instance, knowing how to optimally perform sequences of discrete actions to solve the Tower of Hanoi task) so as to avoid a too long learning process during child-robot interaction. This can be done by setting the learning rate for discrete actions α_Q to zero after pretraining and fixing the inverse temperature $\beta(s)$ to a very high value. This way, the algorithm would start the social interaction with an already appropriate behavior to solve the Tower of Hanoi task, while still being able to adapt online the continuous parameters of action. An intermediate application could consist in having a fixed prelearned discrete behavioral policy with $\alpha_Q = 0$ while still enabling a dynamic tuning of $\beta(s)$ to produce transient variability in the behavior. In the simulations we have shown for Experiment 3, we nevertheless showed that the robot could learn all in parallel. But it's important to highlight the other possible applications of the algorithm. Finally and conversely, one may prefer to freeze the continuous parameters of action (by setting $\alpha_C = 0$, $\alpha_A = 0$ and $\sigma(s, a) \approx 0$) to a level of expressivity and a speed of movement which have already been learned and seem appropriate for child-robot interaction, while still enabling the robot to learn discrete behavioral policies for new tasks. In this context, freezing the contin-

uous action parameters but still permitting active exploration around these prelearned values by letting $\sigma(s, a)$ be dynamically modulated could permit more fluid variability in the robot behavior while still remaining around a set of fixed reasonable averages when sufficient prior knowledge enable to initialize these averages. These different variants of the algorithm offer a wide range of possible applications to human-robot interaction, depending on the level of prior human knowledge and the desired degree of variability.

Supplementary data

We present in the supplementary material the results obtained by the previous version of the algorithm [16] with global exploration rates β_t and σ_t on the multi-state task, to show that it performs less well than the new version. This justifies the extension we made to have state-specific discrete exploration rates $\beta_t(s)$, as well as (state,action)-specific continuous exploration rates $\sigma_t(s, a)$.

We moreover present two simulation videos. Suppl. Video 1 illustrates a simulation of Experiment 3 including different task phases: post-training, changepoints and relearning from scratch. Suppl. Video 2 illustrates a simulation of Experiment 4 where the algorithm is successively interacting with three different children.

Acknowledgement: This research work has been partially supported by the EU-funded Project BabyRobot (H2020-ICT-24-2015, grant agreement no. 687831), the Agence Nationale de la Recherche (ANR-12-CORD-0030 Roboergosum Project and ANR-11-IDEX-0004-02 Sorbonne-Universités SU-15-R-PERSU-14 Robot Parallearning Project), Labex SMART (ANR-11-LABX-65 Online Budgeted Learning Project), the Centre National de la Recherche Scientifique (Mission pour l'Interdisciplinarité ROBAUTISTE Project and PICS no. 279521 SocialRobot Project) and the Royal Society International Exchange Scheme (grant no. IE151293).

References

[1] T. Fong, I. Nourbakhsh, K. Dautenhahn, A survey of socially interactive robots, Robotics and Autonomous Systems, 2003, 42, 143–166

[2] T. Kanda, T. Hirano, D. Eaton, H. Ishiguro, Interactive robots as social partners and peer tutors for children: A field trial, Human-Computer Interaction, 2004, 19(1), 61–84

[3] B. Robins, K. Dautenhahn, R. Te Boekhorst, A. Billard, Robotic assistants in therapy and education of children with autism: Can a small humanoid robot help encourage social interaction skills? Universal Access in the Information Society, 2005, 4(2), 105–120

[4] T. Belpaeme, P. E. Baxter, R. Read, R. Wood, H. Cuayáhuitl, B. Kiefer, et al., Multimodal child-robot interaction: Building social bonds, Journal of Human-Robot Interaction, 2012, 1(2), 33–53

[5] K.-Y. Chin, Z.-W. Hong, Y.-L. Chen, Impact of using an educational robot-based learning system on students motivation in elementary education, IEEE Transactions on Learning Technologies, 2014, 7(4), 333–345

[6] C. Rich, B. Ponsler, A. Holroyd, C. L. Sidner, Recognizing engagement in human-robot interaction, In: 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI), IEEE, 2010, 375–382

[7] S. Ivaldi, S. Lefort, J. Peters, M. Chetouani, J. Provasi, E. Zibetti, Towards engagement models that consider individual factors in HRI: on the relation of extroversion and negative attitude towards robots to gaze and speech during a human-robot assembly task, International Journal of Social Robotics, 2017, 9(1), 63–86

[8] S. Lemaignan, M. Warnier, E.A. Sisbot, A. Clodic, R. Alami, Artificial cognition for social human-robot interaction: An implementation, Artificial Intelligence, 2017, 247, 45–69

[9] C. L. Sidner, C. Lee, C. D. Kidd, N. Lesh, C. Rich, Explorations in engagement for humans and robots, Artificial Intelligence, 2005, 166(1–2), 140–164

[10] S. M. Anzalone, S. Boucenna, S. Ivaldi, M. Chetouani, Evaluating the engagement with social robots, International Journal of Social Robotics, 2015, 7(4), 465–478

[11] M. Khamassi, S. Lallée, P. Enel, E. Procyk, P. F. Dominey, Robot cognitive control with a neurophysiologically inspired reinforcement learning model, Frontiers in Neurorobotics, 2011, 5, 1

[12] J. Kober, J. Peters, Policy search for motor primitives in robotics, Machine Learning, 2011, 84, 171–203

[13] F. Stulp, O. Sigaud, Robot skill learning: From reinforcement learning to evolution strategies, Paladyn Journal of Behavioral Robotics, 2013, 4(1), 49–61

[14] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, The International Journal of Robotics Research, 2013, 32(11), 1238–1274

[15] M. Khamassi, G. Velentzas, T. Tsitsimis, C. Tzafestas, Active exploration and parameterized reinforcement learning applied to a simulated human-robot interaction task, In: 2017 First IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 2017, 28–35

[16] M. Khamassi, G. Velentzas, T. Tsitsimis, C. Tzafestas, Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning, IEEE Transactions on Cognitive and Developmental Systems, 2018 (in press)

[17] W. Masson, P. Ranchod, G. Konidaris, Reinforcement learning with parameterized actions, In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), 2016

[18] M. Hausknecht, P. Stone, Deep reinforcement learning in parameterized action space, In: International Conference on Learning Representations (ICLR 2016), 2016

[19] J. Schmidhuber, Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts, Connection Science,

2006, 18(2), 173–187
[20] A. Baranes, P.-Y. Oudeyer, Active learning of inverse models with intrinsically motivated goal exploration in robots, Robotics and Autonomous Systems, 2013, 61(1), 49–73
[21] C. Moulin-Frier, P.-Y. Oudeyer, Exploration strategies in developmental robotics: a unified probabilistic framework, In: 2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL), IEEE, 2013, 1–6
[22] F. C. Y. Benureau, P.-Y. Oudeyer, Behavioral diversity generation in autonomous exploration through reuse of past experience, Frontiers in Robotics and AI, 2016, 3
[23] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, et al., Learning to reinforcement learn, 2016, arXiv:1611.05763
[24] N. Schweighofer, K. Doya, Meta-learning in reinforcement learning, Neural Networks, 2003, 16(1), 5–9
[25] K. Doya, Metalearning and neuromodulation, Neural Networks, 2002, 15(4-6), 495–506
[26] G. Velentzas, C. Tzafestas, M. Khamassi, Bio-inspired meta-learning for active exploration during non-stationary multi-armed bandit tasks, In: IEEE Intelligent Systems Conference 2017, London, UK, 2017
[27] A. Garivier, E. Moulines, On upper-confidence bound policies for non-stationary bandit problems, 2008, arXiv:0805.3415
[28] H. van Hasselt, M. Wiering, Reinforcement learning in continuous action spaces, In: IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007, 272–279
[29] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998
[30] L. Schilbach, M. Wilms, S. B. Eickhoff, S. Romanzetti, R. Tepest, G. Bente, N. J. Shah, G. R. Fink, K. Vogeley, Minds made for sharing: Initiating joint attention recruits reward-related neurocircuitry, Journal of Cognitive Neuroscience, 2010, 22(12), 2702–2715