Integrated Computer-Aided Engineering 30 (2023) 205–222
DOI 10.3233/ICA-230707
IOS Press



# Using perceptual classes to dream policies in open-ended learning robotics

Alejandro Romero[a], Blaz Meden[b], Francisco Bellas[a] and Richard J. Duro[a,*]
[a]*Integrated Group for Engineering Research, Centre for Information and Communications Technology Research, Universidade da Coruña, Coruña, Spain*
[b]*Computer Vision Lab, University of Ljubljana, Ljubljana, Slovenia*

**Abstract.** Achieving Lifelong Open-ended Learning Autonomy (LOLA) is a key challenge in the field of robotics to advance to a new level of intelligent response. Robots should be capable of discovering goals and learn skills in specific domains that permit achieving the general objectives the designer establishes for them. In addition, robots should reuse previously learnt knowledge in different domains to facilitate learning and adaptation in new ones. To this end, cognitive architectures have arisen which encompass different components to support LOLA. A key feature of these architectures is to implement a proper balance between deliberative and reactive processes that allows for efficient real time operation and knowledge acquisition, but this is still an open issue. First, objectives must be defined in a domain-independent representation that allows for the autonomous determination of domain-dependent goals. Second, as no explicit reward function is available, a method to determine expected utility must also be developed. Finally, policy learning may happen in an internal deliberative scale (dreaming), so it is necessary to provide an efficient way to infer relevant and reliable data for dreaming to be meaningful. The first two aspects have already been addressed in the realm of the e-MDB cognitive architecture. For the third one, this work proposes Perceptual Classes (P-nodes) as a metacognitive structure that permits generating relevant "dreamt" data points that allow creating "imagined" trajectories for deliberative policy learning in a very efficient way. The proposed structure has been tested by means of an experiment with a real robot in LOLA settings, where it has been shown how policy dreaming is possible in such a challenging realm.

## 1. Introduction

A truly autonomous robot should be able to find out on its own how the domains it faces work and what to do to achieve its purpose in them. In other words, throughout its lifetime, the robot should be capable of finding goals that lead to the completion of the functions the designer or user seeks from it learning skills that permit achieving those goals in whatever domains it operates. This is a step up from more traditional artificialintelligence (AI) based approaches where a single function or type of domain is addressed [1,2]. Thus we say that the robot must be endowed with open-ended learning autonomy (OLA) [3]. Furthermore, for the sake of efficiency, we would expect these robots to transfer and adapt knowledge learnt in previously experienced domains to make learning in new domains more efficient. In other words, autonomous robots should be capable of lifelong learning [4]. Joining these two requirements, we would be seeking robots with lifelong open-ended learning autonomy (LOLA) to continue advancing the field. This is a very challenging objective that has partially been addressed from many perspectives with some remarkable results.

LOLA implies not only learning new skills or models when a goal is provided by a designer, such as in Reinforcement Learning (RL) [5] or other recent approaches [6]. It also implies discovering goals compatible with the functions assigned to the robot independently of the domain it finds itself in, as well as managing all the knowledge that is acquired in the process. This knowledge includes motivational knowledge, perceptual classification knowledge, modeling knowledge or skill related knowledge among others Unlike more

*Corresponding author: Richard J. Duro, Escola Politécnica de Enxeñaría de Ferrol, Campus de Esteiro, 15403, Ferrol, A Coruña, Spain. E-mail: richard@udc.es.

ISSN 1069-2509 

standard AI and machine learning approaches [7–10] that usually concentrate on one skill and are generally based on off-line learning, this knowledge must be learnt through on-line interaction in continuous, uncertain, and dynamic domains These types of systems must also contemplate the capability of transferring knowledge [11] in an orderly and efficient manner. Consequently, an internal management structure in the form of some type of cognitive architecture must be established to provide for: a) open-ended learning, in this case providing components concerned with motivational and knowledge modelling aspects and b) lifelong learning, creating components devoted to contextual storage and knowledge transfer and reuse aspects.

Cognitive architectures thus make up a formal structure to achieve LOLA in robots. From a functional perspective, these architectures provide a software framework to support and relate the knowledge designed by the robot creator, as well as that discovered and learned by the robot itself so that useful decisions can be made. In addition, such a structure must be able to adapt its decision-making processes to cater to its level of knowledge elaboration. That is, a key property of cognitive architectures is to properly implement and complement deliberative and reactive (intuitive) decision-making processes, so that proper LOLA operation can be accomplished by the robot.

### *1.1. Decision-making processes in LOLA*

As indicated above, to achieve optimal action selection in a robot there is a need for both reactive and deliberative processes. A correct equilibrium between them under different circumstances is the key to successful robotic behaviors in real operation [12].

Deliberative capabilities permit deciding on action through a prospection/evaluation cycle following different possible action paths. They are necessary if a robot has no experience in a domain or is faced with novel circumstances. Deliberation permits selecting an action without actually having learnt a policy. It is thus a way for the robot to explore and gain experience for the generation of new knowledge. Thus, the robot must determine the best action towards the achievement of its objectives from its starting state. Consequently, appropriate world models (WM), also called state-transition models in RL, should be available to perform prospection; whereas utility models (UM) are needed to be able to evaluate states. However, in LOLA settings WMs and UMs are not known beforehand, so the cognitive architecture must also support their online learning.

In the case of the autonomous acquisition of WMs, several approaches have been recently proposed in the field of Intrinsically Motivated Open-ended Learning (IMOL) [13,14]. These approaches make use of intrinsic motivations (IM) to induce agents to reduce the discrepancy between their knowledge and their perceptions [15]. Some of the most influential work, such as those carried out in [16,17] on machine learning or in [18,19] on robots, have led to algorithms that allow autonomous agents to learn models of the domain in which they operate, so that they can "predict" and anticipate the environmental effects of their actions.

Regarding UMs, different ways of acquiring and refining them have been proposed in the literature [20]. One of the most popular approaches in the field of RL is through the concept of value [21,22] and the definition of value functions (VF). These functions generate an accurate expected value for each point in the state space, i.e., they provide the utility of each point towards the achievement of a goal. Thus, they provide an indication of how far away the goal is, allowing for a quantitative comparison of different possible trajectories towards it.

These concepts have also been applied under LOLA conditions, where different procedures have been developed to learn VFs online. An example in this line is that by [23,24] who have presented an approach based on the use of eligibility trace strategies to construct VFs when working in continuous domains that are not known in advance.

Reactive or intuitive capabilities, on the other hand, are useful, for instance, when the robot is already competent and requires acting speedily due to real-time operational constraints imposed by dynamic environments. In this case, the robot should select an appropriate policy that outputs the optimal action to be applied as a function of the goal it seeks and its current state. The policies the robot chooses must have been previously designed in by a designer or they should have been previously learned by the robot itself. Several approaches have been proposed for learning policies in traditional robotics and in the field of RL. These go from Q-learning [25], policy gradient [26], and actor-critic methods [27] to model-free policy search [28] and many others. However, they are mainly aimed at pre-obtaining the policy given a pre-established reward function over the whole state space, with some remarkable exceptions like [29].

Figure 1 contains a diagram of the decision-making processes in LOLA, with the main cognitive blocks involved. Label “a. Deliberative” shows the typical execution path where, given a current state, for each action

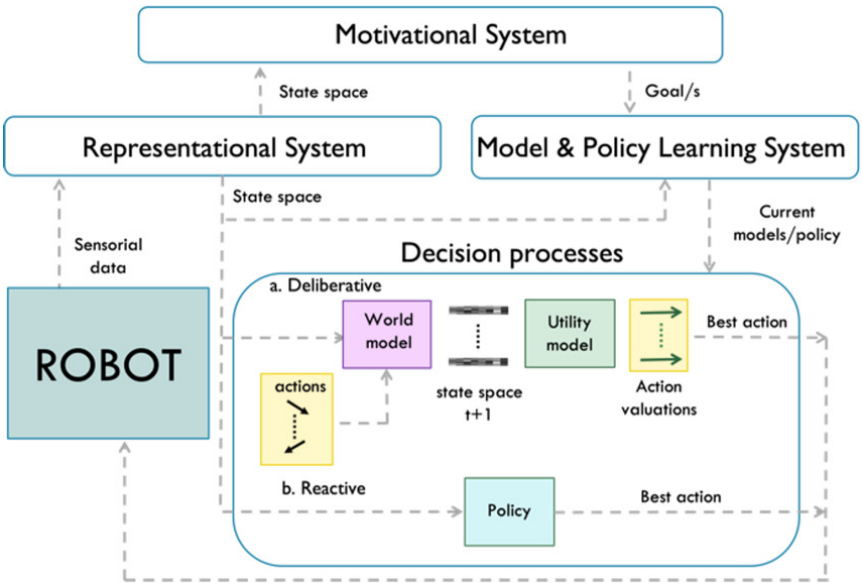


Fig. 1. Diagram of the decision-making processes involved in LOLA.

of the candidate set, the WM predictions are used as input in the UM to evaluate them, so the best one can be selected. On the other hand, "b. Reactive" shows a single policy that directly provides the best action. It is the correct balance between one branch and the other what is really challenging in this scope.

Cognitive architectures that support LOLA must support both deliberative and reactive decision processes. For this very reason, and as shown in the "Model & Policy Learning System" block of Fig. 1, they must allow WM, UM and policy learning. This learning must take place while the robot interacts with the world (not beforehand) and without any preset reward function over the entire state space. In fact, in LOLA environments, the reward that can be obtained in a domain that is new to the robot has to be discovered as the robot interacts with it. Moreover, depending on the robot's activity in that domain, it usually does not cover the entire domain, so knowledge about the reward function is relative and imperfect.

WM or UM learning, as commented before, are relatively straightforward to transform into online processes. This is because the data they require for learning is directly accessible through interaction with the environment. They just need states, actions performed and rewards in different instants of time, no matter what the robot is doing or trying to achieve. As some authors posit [30], they may even be random.

Policies, on the other hand, involve a mapping between the states the robot may find itself in and the action to be carried out to be able to achieve a specific goal. Consequently, given this goal dependency, they are a bit more difficult to learn on-line The main problem is that to learn policies it is necessary to be able to predict or test the effects of the robot's actions as related to the goal to be achieved. Moreover, this prediction must be possible starting from any point in the state space that is relevant. Thus, to perform this optimization the robot must either try out all the possible actions over all its real-world states (with the cost and danger this may entail) or it needs to internally perform a deliberative process using an appropriate WM and some type of UM that allows evaluating which is the most adequate action to move towards the goal.

Given the problems posed by the first approach, some authors in the RL literature have already hinted towards the second strategy for policy learning in robots that are operating online. They usually refer to it as *dreaming, hallucination*, or *imagination*, in the sense that the policy learning processes rely on internal reasoning and deliberation, as a sort of "introspection". In [30] the authors present an experiment where they learn a policy by means of a so-called dreaming process in order to control a real robot. This dreaming process makes use of a learned WM that is used to determine trajectories to the goal. Therefore, they learn a policy by interacting with the WM instead of the real world. Similarly, in [31] the authors present an agent that learns behaviors to solve some visual control tasks. These behaviors are learned by backpropagating gradients of learned state

values through imagined trajectories in the state space given by a WM, which is also learned.

However, in these RL approaches, the learning processes are possible because the goals are known in advance so perfectly defined UMs in the form of VFs can be pre-learned This means that the data required for learning the VF is well-defined, and it covers the complete domain Therefore when learning a policy using this approach, any state is a possible valid input state.

In the situations contemplated by LOLA robots, this is not usually the case. The goals the robot must achieve throughout its lifetime are not known beforehand. In fact, they have to be discovered through interaction with the world. This implies that their corresponding UMs must be created on the fly as a function of what the robot needs to achieve each moment in time. No explicit reward function is available and reward arises from the satisfaction of internal robot motivations (as will be detailed in Section 2). To make it even more challenging, the cognitive architecture must learn such UMs concurrently with the discovery of new goals and in different domains that could change over the lifetime of the robot Consequently, data for model learning could be obtained in an interspersed fashion, with different learning periods corresponding to different goals. Hence, to cope with all the uncertainties existing in LOLA settings, a new approach for policy learning must be developed.

### *1.2. Specific scope and contributions*

Endowing a cognitive architecture with complete decision-making capabilities (deliberative and reactive), implies solving the challenging problem of policy learning in LOLA settings. To this end, there are three main issues to be addressed.

Firstly, motivations must be defined in a domain-independent representation This allows for the autonomous determination of domain-dependent goals since we deal with unpredictable domain changes.

Secondly a method to determine expected utility under the challenging and uncertain circumstances of LOLA must be developed, so that a continuous version of UMs can be inferred. This is required because, in the most challenging case, reward or utility can only be obtained when in the state space point or area corresponding to the goal.

Finally, it is necessary to provide an efficient way to internally generate data on which perceptual states are relevant for policy learning. This is because policies do not have to be valid in the whole state space, but only in areas where WMs and UMs are well defined (since policies are learned from them by "dreaming").

The first two issues have been previously addressed in the scope of the e-MDB cognitive architecture [32]. First, a domain-independent strategy for goal creation, based on needs and drives, has been developed and tested in real experiments with robots [33]. Second, a method for online value function learning based on predicted utility and Separable Utility Regions has been also studied in detail [23].

Dealing with the third issue established above makes up the main scientific contribution of the current work. Thus, here we propose the introduction of a metacognitive structure within the cognitive architecture that represents Perceptual Classes, which will be represented within the architecture by components called P-nodes These components, which are contextually linked to goals, as well as to WMs and UMs, are acquired and delimited during the on-line operation of the robot as it learns these models. Once learnt, they permit inferring the relevant perceptual domain from which the robot is able to achieve the goal. Thus, they can be used to generate "dreamt" or "imagined" starting state space points that allow generating "imagined" trajectories for deliberative policy learning in a very efficient way.

With these three methods implemented in a cognitive architecture, policy dreaming in LOLA settings is possible, as will be shown here by means of the execution of an experiment with a real robot. In it, a procedure for online policy learning based on neuroevolution, which uses the P-nodes as a core element for efficient data generation in internal deliberation processes (dreaming) will be tested.

The rest of the paper is structured as follows. In Section 2, a formal statement of the policy dreaming problem in LOLA setting is presented, based on a comparison with the same problem in traditional RL. Section 3 contains a brief description of the motivational system that allows for goal creation and UM learning. In Section 4, Perceptual Classes are described in detail, and the specific policy dreaming strategy based on neuroevolution is explained. Section 5 presents the experiments run to show the validity of the proposed approach. Finally, Section 6 describes the conclusions, and open issues that need to be addressed in future work.

## 2. Policy learning by dreaming in LOLA

Figure 2 left shows a schematic representation of the processes involved in policy learning and decision

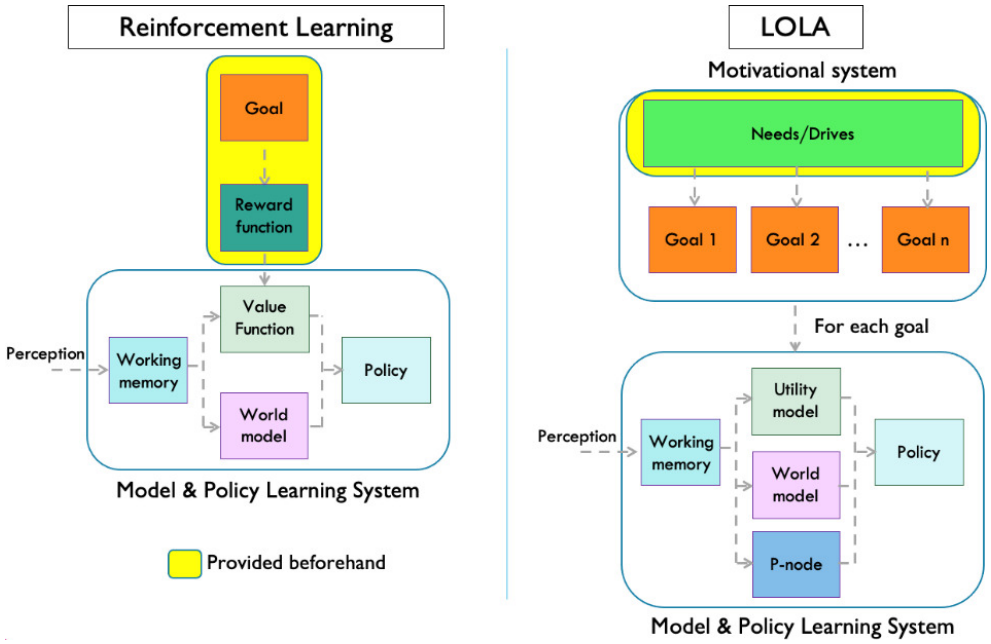


Fig. 2. Schematic representation of the processes involved in online policy learning for a robot in traditional RL settings and in LOLA settings. The goal represented in the figure are operational, so they are related with the task to be solved in specific domains.

making following the "dreaming" trends in RL, such as those used in [31,34]. In these contexts, the authors establish a clear goal to be achieved through the instantiation of a reward function, which defines, within the state space, all the rewards that are possible for the domain. This is highlighted in the figure in a yellow block.

The "Model & Policy Learning System" block receives episodes consisting of actions performed and state space points in different instants of time, as well as the reward, obtained as the robot interacts with the environment (represented using the "perception" label in Fig. 2). These episodes are stored in a Working memory and used to train world models and value functions (VF) online.

Once these models have been learnt, they are used in a deliberative process to "dream" different trajectories of the robot towards the goal, and with them, learn a policy online, in a similar way to that shown in Fig. 1 in the "a. Deliberative" branch. Thus, different sequences of actions are generated using the WM to predict their outcomes. These outcomes in turn are evaluated using the VF, allowing to choose the best actions with respect to the goal that has been set by the designer. These trajectories usually start from random points in the state-space and follow a path that can always be evaluated by the reward function, which was provided beforehand and covers the whole state space. Consequently, any state space point evaluated will result in a coherent value.

The previous learning process is, in general, more complicated and diffuse for LOLA (see Fig. 2 right). First, in LOLA some processes run concurrently with others because there are different domains that are visited in an intermingled fashion, so separate learning stages are not possible. For instance, WM learning could be performed in parallel with representation learning, so that the representational space is optimal for WM learning. Moreover, establishing when a model is reliable in LOLA settings is complicated as, due to lifelong learning processes, it will be improved in different operational stages.

Anyway, the most important differences stem, on the one hand, from the fact that in LOLA a designer does not provide goals for specific domains, as in RL. This is quite an important difference as it implies that goals must be discovered in the different domains, and then UMs have to be created (without a predefined reward function) for each specific domain. Therefore, UM learning is much more challenging. The top block of Fig. 2 right displays the basic elements involved in the motivational system of a LOLA robot with regards to learning and decision making.

As will be explained in Section 3, the approach presented here for goal discovery relies on the concepts of Need and Drive (top block of Fig. 2 right), and in a

balancing strategy that allows to manage the discovered goals, so that robot operation is feasible. Needs and Drives are defined on a generic representation space (motivational space) that is internal to the robot, so they are domain independent. This way, managing goals discovered in different domains is possible. But the goals are still defined in the domain specific representation. Goals are linked to drives, so achieving a goal will lead to the satisfaction of a drive. Therefore the drive is a way to provide an indication of reward. How drives and goals can be connected in online operation is a key aspect of the motivational system because it defines the relation between domain dependent and domain independent representations. A possible implementation was defined and tested in [33].

The second main difference between policy dreaming in RL and LOLA is a consequence of the lack of reward functions that encompass the whole domain. The UMs that are obtained may only be valid or relevant in certain regions of the state-space and only in certain domains. This implies that, when the robot is dreaming, it cannot randomly select state-space points or consider trajectories that are not within the areas where the UM is valid. Otherwise, the evaluations, and thus the results of the deliberation processes, will be unreal or even absurd.

To consider that UMs may not be defined over the whole state space, we will introduce the concept of perceptual class and contextual processing (presented in Section 4.1). They will allow to learn in an interactive way in what areas of the state space the UM is reliable, and even how much. This has been represented in Fig. 2 right by means of the P-node block, a function that must also be learned online, and which is used in the policy dreaming process.

Following this approach, initially the robot explores its domain using an uninformed deliberative process (WMs and UMs are not reliable) that leads to an amount of randomness of action, and learns the representation, WMs and UMs. As these WMs and UMs improve, the robot operation is guided by ever more informed deliberative processes. Once they reach a given reliability level, they can be used together with the associated perceptual classes, in an internal process to learn the policy. Finally, the robot can use the policy, once learnt, in a reactive fashion [35,36].

Summarizing, policy dreaming in LOLA settings goes beyond the state of the art in RL. Under these conditions, UM learning is controlled by a motivational system which runs in a domain-independent representation to support domain specific goal discovery. This makes the process much more abstract and challenging. An additional component that provides an indication of where in state-space the UM is reliable, and even how reliable it is, must be learnt in parallel with the UM. It also must be used in policy dreaming to prevent deceitful deliberative processes. In the following two sections we will describe the approach we propose to address these issues.

## 3. Motivational system

In this section we describe the motivational system that we propose to guide the robot's learning processes when it operates under LOLA conditions. The main objective of this system is to allow the robot to discover goals, learn how to reach them consistently, and select which ones are active in the different domains. Therefore, it is the driver of the robot's open-ended learning operation, as it provides domain independent needs and drives [33,37] which are progressively linked to domain specific goals that are in line with the needs of the system. It must be pointed out that all the representations used in this approach are sub-symbolic.

### *3.1. Needs and drives*

Needs correspond to internal states that the robot wants to reach or preserve. Thus, a need $n_j$ is a goal point or area within the motivational state space $M$ of the robot. $M$ is a specific state space that is designed by the robot's designer and its dimensions are given by domain independent variables $m_i$ usually referring to internal parameters of the robot.

Each need $n_j \in M$ has a drive $D_j \in \mathbb{R}$ assigned to it that reflects how far the system is from satisfying $n_j$. The value of the drive $D_j$ is established through a function $f_j$ that provides a distance (not necessarily Euclidean) between the current motivational state of the system $x_t \in M$ and the point $n_j \in M$ where the need is satisfied:

$$D_j = f_j(x_t, n_j)$$

These drive functions $f_j$ are defined by the system designer and permit assigning a purpose to the robot by balancing and prioritizing the different needs within its motivational system.

It is important to note here that the motivational space $M$ is different from the robot operational state space $S$, whose structure depends on the specific representation the robot is using for a particular domain.

The fundamental idea of this motivational approach is that the robot will always strive to satisfy its needs and thus, to reduce the values of its associated drives.

### 3.2. Goals and expected utility

Since robots operate in real domains, to reduce the values of their drives, they must find perceptual situations $S_i \in S$ that, within the particular domain they are currently operating in, result in the satisfaction of the needs in $M$. These perceptual situations are called goals.

Thus, a goal can be defined as a point or area $G_r \in S$ that when achieved leads to the reduction of the value of at least one drive $D_j$ associated to a need $n_j \in M$. Goals provide utility. Thus, for a given domain $k$, the utility ${}^kU$ of a point $S_i \in S$ will be a measure of the change in the value of the corresponding drive $D_j$ when that point is reached. Therefore, by creating a utility model ${}^kUM$, it would be possible to know how close the robot is to satisfying a drive (or to reaching a goal), when at that point $S_i \in S$.

The problem when working in real domains, and therefore with real operational state spaces, is that there are not too many areas that provide utility. In fact, utility often appears sparsely and in a discrete form. Consequently, finding utility can be a very complex problem. To solve this problem and to improve the chances of finding goals (areas that provide utility), an option is to model the state space utility in terms of expected utility. Thus, the expected utility, ${}^k_r\hat{U}_i$ with respect to a goal $G_r \in S$ of a point $S_i \in S$ can be defined as the probability of reaching $G_r$ starting from that point modulated by the utility provided by the goal. Thus, an expected utility model, ${}^k_r\widehat{UM}$, could be constructed for each goal, $G_r$, by determining ${}^k_r\hat{U}_i$ with regards to that goal for each point $S_i \in S$. A remarkable feature of these models is that the expected utility value increases monotonously as we approach $G_r$. Therefore, by constructing this type of models, the robot has a mechanism to choose actions that lead to the goal in a deliberative manner. For compactness reasons, in the rest of the article we will refer to the expected utility models simply as utility models, *UM*.

### 3.3. Types of needs and drives

The only control the designer has over the behavior of the robot is by defining its needs and creating the appropriate drives. Therefore, when needs and drives are created to define the behavior of a robot, it must be endowed with the necessary tools to explore the domains, find goals, and learn how to achieve them, i.e., model expected utility. These objectives lead to a classification of needs and drives into two different groups: operational and cognitive needs and drives.

Operational needs and drives $({}_{op}n_j, {}_{op}D_j)$ are associated with the purpose that must be instilled in the robot. They reflect what the designer wants the robot to do, but in a domain-independent manner. Thus, for each domain the robot faces, it must identify and learn to achieve goals to fulfill these operational needs.

Cognitive needs and drives $({}_{cg}n_j, {}_{cg}D_j)$, on the other hand, enable the robot to efficiently explore and acquire relevant knowledge. These drives reflect how the designer wants the system to regulate its learning process. Therefore, their satisfaction may be related to how much is explored (e.g. novelty [38]), to obtaining WMs (e.g. curiosity [39]), to generating goals by seeking to find ways to produce effects over the environment (e.g. effectance [40]), or to acquiring competence over those goals (e.g. competence [41]).

The details of how to design these drives have been discussed in [33,42]. What is important here is the idea that through the correct selection and design of drives a robot can acquire the ability to find goals in the domains it faces and provide motivation for modelling the domains as well as learning how to achieve those goals consistently.

## 4. Autonomous policy learning

As mentioned in the introduction, when learning policies under LOLA conditions, it is essential to know the context of operation of the robot. That is, to learn a policy it is necessary to know which goal to achieve and in which domain, but it is also fundamental to determine under which perceptual conditions it is possible to do it. In this case, it cannot be assumed that the knowledge elements that are available (i.e., world and utility models) are valid throughout the complete state space of the robot for that domain.

Consequently, once the motivational system presented in Section 3 allows discovering goals and learning the UMs necessary to reach them in a deliberative manner, to be able to learn a policy from that knowledge it is still necessary to solve the problem of determining under what perceptual situations the UM is valid so that it is not used outside its "universe". To address this issue, as commented in Section 2, we describe here the concept of perceptual classes and P-nodes.

### 4.1. Perceptual classes and P-nodes

We define a perceptual class, $\hat{S}_j$, as an area of the perceptual/statespace whose points share some operational

characteristics and are thus taken as a single entity. This leads to a discretization of the perceptual/state space into areas whose points lead to the same results under specific contexts. When considering action determination, this means that the response of the robot in terms of action is the same for every point in the perceptual class when operating in the same domain and striving to reach the same goal. Thus, a perceptual class can be considered an abstraction of perceptions leading to a discrete, higher-level representation associated, in this case, to a particular response of the robot.

To represent perceptual classes within a cognitive architecture, a knowledge component called perceptual node or P-node was proposed in [43]. A P-node, denoted as $P_j$, is a component that is activated when a perceptual state $S_i$ belongs to a given perceptual class $\hat{S}_j (P_j = 1 \leftrightarrow S_i \in \hat{S}_j)$. Thus, a P-node is a metacognitive component in the form of a filter over the perceptual state space. It permits determining the presence of state space points that belong to $\hat{S}_j$.

Under ideal conditions, a P-node $P_j$ will be perfectly active when a perception $S_i$ belongs to its corresponding perceptual class $\hat{S}_j$. However, in more uncertain settings, such as the real world, a probabilistic representation of P-node activation may be considered. In this representation the activation level depends on how confident the system is about $S_i \in \hat{S}_j$. This confidence function in the perceptual space, denoted as $\Gamma_j(S_i)$, can be created using the information the robot has about the points it has experienced. Thus, the membership to a perceptual class $\hat{S}_j$ of a state represented by P-node $P_j$ is given by:

$$P_j = \Gamma_j(S_i) p_j(S_i)$$

where $p_j$ denotes the activation of the P-node under ideal conditions. To compute this confidence function, we make use in this article of a point-based clustering representation [43], although neural network based representations would also work, as the authors have described in [44].

In a point-based interpolation approach, the state space area represented by a P-node is given by the set of points actually perceived by the robot and some distance-based rules. This way it is possible to delimit the areas for which the system hypothesizes the P-node is active. In this line, state space points can be classified into two categories: points for which the P-node should be active (activation $= 1$), which will be called simply *points*, and points for which the P-node should be inhibited (activation $= 0$), which will be called *anti-points*. Both types of points are stored in a memory structure related to the P-node. All other points within the class, which were not actually experienced by the robot, will be assigned confidence values in the interval [0:1] depending on their distance to *points* and *anti-points*. The specific algorithm used here for learning the perceptual class that corresponds to a P-node is described in [43]. Algorithm 1 describes the procedure to decide on the activation of a P-node representing a perceptual class $\hat{S}_j$ and given a new perceptual situation $S_i$. This procedure leads to the delimitation of an area in the state space that activates the P-node. Through continuous interaction with the environment, new P-nodes will be created, and existing P-nodes will be populated with new *points* and *anti-points*, so that perceptual classes will be progressively delimited.

Therefore, P-nodes are very important for determining the perceptual context when working in high-dimensional continuous domains, such as those that a real robot may encounter. So, by associating these nodes to the goals to be achieved, a high-dimensional delimitation of the perceptual state space can be established. This will make it possible to know more accurately under which perceptual conditions these goals are achievable and also to delimit those perceptual situations that need to be considered when learning a policy.

**Algorithm 1:** P-node activation algorithm

$P_j = \{p_1, \ldots, p_n, a_1, \ldots, a_n\}$ P-node given by points and anti-points
$P_j(x)$ denotes the activation of $P_j$ for perception $x$
$S_i$ is a new perception

```
Calculate k as the closest point/anti-point to S_i out of those representing P_j
if (k ∈ {p_1, ..., p_n}) then
    P_j(S_i) = 1/(||k − S_i|| + 1)
else if k ∈ {a_1, ..., a_n} then
    Calculate the centroid of {p_1, ..., p_n} as C
    if (||C − S_i|| < ||C − k||) then
        Calculate k' as the closest point to S_i out of those representing P_j
        P_j(S_i) = 1/(||k' − S_i|| + 1)
    else
        P_j(S_i) = −1.0
    end if
end if
if (P_j(S_i) < ε) then
    P_j(S_i) = 0.0
end if
```

### 4.2. Policy dreaming strategy

As mentioned in Section 1, given that in open-ended learning problems the domains and tasks to be addressed by the robot are not known at design time, pre-learning policies directly is not possible. However, this

learning can be achieved using the deliberative models (UMs and WMs) and the perceptual classes that the robot has acquired on-line, as detailed in Section 2. Once they have been adequately learnt, these models contain the information needed to be able to decide on the optimal action to apply at each moment in time in order to reach a goal in a particular domain. They permit the implementation of prospection processes (using WMs), and the later evaluation of resulting states (using UMs). However, to take into account the areas where these models are valid, that is, the areas of the perceptual state space from which it is possible to achieve these goals, their associated perceptual classes (P-nodes) need to be known.

As previously indicated, the main objective of a robot with a motivational system, such as the one presented in Section 3, is the satisfaction of its needs. That is, the maximization of drive satisfaction. As indicated above, this can be achieved by maximizing expected utility $\hat{u}$ in the particular domain the robot is operating each instant of time. In other words, we are seeking a policy that results in actions leading to the maximum expected utility for a given domain or sets of domains.

In our approach, the different components (models and policies) have been implemented by means of Artificial Neural Networks (ANN). This is so because in our cognitive architecture, the e-MDB [32], knowledge is represented using neural networks. Therefore, we will address policy learning as the process of producing an ANN that implements the policy. In addition, we must consider that it is not possible to estimate beforehand how complex the task to be solved will be, which makes it difficult to decide on the appropriate ANN size. Thus, model and policy learning will be achieved by means of a neuroevolutionary algorithm. In particular, we use the NEAT [45] algorithm, that allows learning the network architecture in terms of number layers and of neurons per layer, as well as which layers connect to which, and the weights of the individual connections. Thus, it is not necessary to establish beforehand the architecture of the network.

As explained in Section 2, online policy learning starts once a reliable WM and UM have been learned in a given domain. At this point, the process begins, like any other evolutionary algorithm, by generating a population of candidate policies ($\Pi$) in the form of random ANNs. These policies are evolved taking the world and utility models as "simulators and evaluators of the current reality". Additionally, the P-node information is used to generate the initial states from which deliberation starts, since it indicates for which perceptual states the UM is applicable. Algorithm 2 describes the procedure used to evaluate an individual policy $\pi_i$ from the policy population $\Pi$, starting from a set of perceptions/states that are extracted from P-node $\mathrm{P}^* \in \mathrm{P}_n$ to make them representative.

**Algorithm 2:** Policy evaluation process

$p_k$: current perceptual point
$P^*$: set of representative perceptions from the P-node $P_n$
$\pi_i$: individual policy
$\Pi$: population of candidate policies
$\pi^*$: optimal policy from $\pi_i$
*WM*: world model
*UM*: utility model
$\hat{u}_k$: expected utility provided by UM for $p_k$
$\bar{u}_i$: average expected utility for $\pi_i$
$a_{ik}$: action provided by $\pi_i$ for $p_k$

**for** $\pi_i \in \Pi$ **do**
  **for** $p_k \in P^*$ **do**
    $a_{ik} \leftarrow \pi_i(p_k)$
    $p_{k+1} \leftarrow WM(a_{ik}, p_k)$
    $\hat{u}_{k+1} \leftarrow UM(p_{k+1})$
    $\bar{u}_i \leftarrow getAverage(\hat{u}_{k+1})$
$\pi^* \leftarrow \mathrm{argmax}(\bar{u}_i)$

This process is repeated for a relevant number of representative points of $P^*$, $p_k \in P^*$ averaging the expected utility, which is used as the evolutionary fitness for policy $\pi_i$. The same set of perceptions and world and utility models are employed for the evaluation of the whole policy population. This way, the best policy (the one with highest average expected utility) at the end of evolution, is the one that the robot will use to achieve that specific goal in that domain and under those specific perceptual conditions. Therefore, for each domain in which the robot operates, and whatever goals it discovers, it will have a method that will allow it to acquire the necessary policies to achieve them.

To sum up, it must be pointed out that unlike in the cases presented in [31,34] related to more traditional RL settings, where it was assumed that models were valid throughout the whole state space, P-nodes are key elements in policy learning under LOLA settings. In the case of LOLA, goals have to be discovered and models are acquired as the robot interacts with the world. There is no guarantee, or even need, that they are valid in the whole state space. Consequently, to be able to run dream like policy learning processes that do not go into unmodeled areas of state space, and thus produce absurd predictions or evaluations, the area of validity of the models must be tracked and used when dreaming. This is precisely what is done through the use of P-nodes, hence their importance.

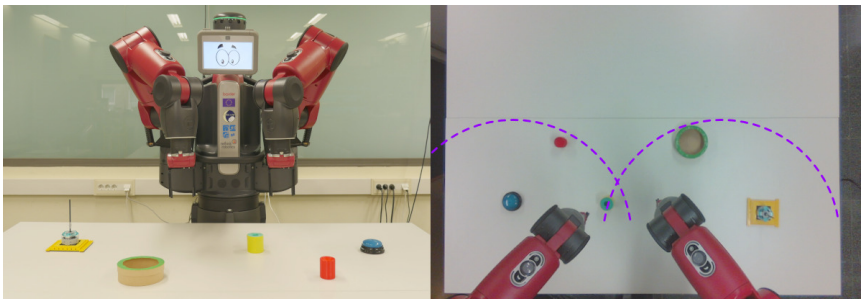

Fig. 3. Scenario (left) and reachable areas (right). The dashed lines display the reachable area for each arm.

## 5. Experiments and analysis

The objective of this section is to provide a real example of how the procedures commented above make it possible to learn policies when operating under conditions of lifelong open-ended learning autonomy, that is, when the goals to be achieved and the domains of operation are not known beforehand. Therefore, we will show how through the appropriate design of the motivational system of a robot it is possible to discover goals in the different domains and learn how to reach them in a deliberative way. In addition, we will see how from this knowledge it is possible to learn reactive behaviors in the form of policies through an offline dreaming process mediated by P-nodes representing perceptual classes. We will also address the creation and management of perceptual classes since they are one of the key elements of the proposed method.

### 5.1. Experimental setup

An experimental setup has been considered in which a real robot must learn to organize a set of objects located on a table. Figure 3 shows this setup, which includes a Baxter robot, a white table, a circular box, a base with a stick for stacking objects, a push button, and some movable elements: a solid red cylinder and a yellow cylinder with an axial hole. The final purpose of the robot is to pick up the red cylinder and place it in the box and insert the hollow cylinder in its base. The robot must determine that this is what it needs to do and learn to do it irrespective of the starting object positions on the table. Additionally, the box in which to deposit the solid cylinder may start out of the reach of the robot. In this case, to bring it closer, the robot will have to press the push-button first. The right image in Fig. 3 shows the areas that can be reached by the robot arms.

To achieve this response, the motivational system of the robot is the only element that has been designed beforehand. We have endowed it with four different needs and their corresponding drives (one operational and three cognitive ones). For each drive we have created an associated drive function.

The operational drive that has been defined is related to object collection. This drive, ${}_{op}D_{task}$, is associated with a need ${}_{op}n_{task}$ that is linked to a sensor that detects whether the objects are in their corresponding storage locations. Thus, each cylinder placed at its destination will produce partial satisfaction of the drive. Similarly, the drive will be fully satiated (the need achieved) when the table is completely organized (both cylinders are in their corresponding storage places). As it is an open-ended learning problem, the robot does not initially know how to satisfy its need. That is, it does not know that correctly storing the objects is what it needs to do. Consequently, it must first figure this out.

As explained in Section 3, to foster exploration, goal discovery and skill learning, the robot is endowed with a set of cognitive needs and drives based on novelty, effectance and competence. As this is not the main goal of the paper, we will not go into details of its implementation, nonetheless, a detailed explanation of it can be found in [33]. These drives (operational and cognitive) will be the ones in charge of allowing the robot to learn to grasp the cylinders and place them in their corresponding storage locations. They will guide the learning of the different deliberative models and P-nodes.

Concerning the perceptual system, $P \in \mathbb{R}$, in this experiment we use an RGB-D camera located on the ceiling of the room and binary sensors associated to the presence of objects in the grippers of the robot and to button pressing. The camera information is redescribed in the form of distances between the objects and the robot's end effectors, and relative angles between the objects and the robot base. Specifically, the perceptual vector has 18 dimensions: 2 binary variables corresponding to the gripper sensors, 1 binary variable re-

Table 1
Parameters of the evolutionary process

| | |
|---|---|
| Generations (epochs) | 1000 |
| Candidate policies (population) | 100 |
| Input neurons | 18 |
| Output neurons | 4 or 2 |
| Number of perceptions/states used for evaluation | 1000 points evaluated in batches of 100. Batches changed every 30 generations. |

lated to the button state, 10 continuous variables related to distances from each robot arm to each of the 5 objects shown in Fig. 3 ($d_{object\text{-}arm} \in [0\text{: }1]$), and finally 5 angles corresponding to the orientation of these objects with respect to the robot's body ($a_{object} \in [-1\text{: }1]$):

$$P(t) = \begin{pmatrix} gripper_1, gripper_2, button \\ d_{red\text{-}r}, d_{red\text{-}l}, a_{red}, \\ d_{green\text{-}r}, d_{green\text{-}l}, a_{green}, \\ d_{box\text{-}r}, d_{box\text{-}l}, a_{box}, \\ d_{base\text{-}r}, d_{base\text{-}l}, a_{base}, \\ d_{button\text{-}r}, d_{button\text{-}l}, a_{button} \end{pmatrix}$$

Regarding action space, $A \in \mathbb{R}$, the Baxter robot presents a two-arm configuration, each offers 7 degrees of freedom. To carry out the experiment, and for the sake of clarity, we have defined a set of actions that control the motion of both arms, i.e., the change in the direction of movement of the arm end effector in sexagesimal degrees ($\alpha \in [0\text{: }360]$) and its displacement in cm ($v \in [0\text{: }5]$). These actions are continuous in the ranges indicated. Therefore:

$$A(t) = (\alpha_{left}, v_{left}, \alpha_{right}, v_{right})$$

where $\alpha_{left}$ and $v_{left}$ are the direction of movement and the displacement for the left gripper, and $\alpha_{right}$ and $v_{right}$ are the direction of movement and the displacement for the right one. The movement of the arms is carried out at a constant height. In addition, the actions related to the opening and closing of the grippers are predefined: if the robot reaches one of the cylinders with one of its grippers, it will automatically pick it up. Also, in the case of reaching the box or the stick when cylinder is in its grasp, it will drop it. Additionally, if both arms cross when one of them has the cylinder in its gripper and the other is empty, the robot will automatically transfer the object to the empty gripper. This action is helpful since the robot is not able to reach the whole table with only one arm.

Finally, the models and policies are implemented using Artificial Neural Networks (ANNs). The MultiNEAT implementation of the NEAT algorithm, extracted from [46], was used to evolve the ANNs corresponding to the policies. The evolutionary process was configured as shown in Table 1. The parameters not included in the table were kept at their default values. These networks have 18 inputs (the perception of the robot at each time instant), and different outputs depending on whether the policies control the movement of one of the arms (2 outputs) or both of them (4 outputs).

### 5.2. Experiment dynamics

Initially, the robot has no idea where the goals are or how to reach them. That is, it does not know the UMs or under which perceptual conditions to use them. Consequently, to achieve its purpose, it must discover its goals and learn the UMs that allow it to reach them on demand. It must also associate them with their corresponding contexts so that goals can be reached regardless of the initial conditions of the scenario. So, for this to happen, it will also have to learn the different P-nodes. We are assuming that the WM representing the domain in which the robot is operating has already been learned in previous stages where the robot interacted freely with the world. This learning is carried out online from the values that are stored in a short-term episodic memory as the robot interacts with the world. An example of this procedure can be found in [35].

The robot will initially explore the environment in the search for novel states or states that provide effectance, which will lead to the discovery and creation of different goals. The creation of goals will also involve the creation of UMs and P-nodes associated to their achievement. Therefore, the robot will alternate between learning the UMs, modelling the activation areas of the P-nodes and exploring its environment in the search for new goals. Thus, initially the operation of the robot will be guided by cognitive drives and deliberative decision processes. However, as soon as the models used for deliberation have been adequately learnt, i.e., the robot consistently reaches the goal from whatever state it is in within the area delimited by the P-node, the dreaming-like learning of the policies can start. From the moment the policy has been learnt, the robot can resort to it as a fast reactive decision process to achieve its goal.

The results of this learning process are shown in Fig. 4. The figure displays the evolution of the time that the robot required to reach its purpose and the moments when the different UMs and P-nodes were considered learned. As a result of this learning process, the robot has discovered six different goals, and has learned their UMs and associated P-nodes. In addition, following

Table 2
Goals discovered after the execution of the experiment

| Goal | Description |
|---|---|
| $G_1$ | Button pressed |
| $G_2$ | Hollow cylinder gripped in left arm |
| $G_3$ | Hollow cylinder gripped in right arm |
| $G_4$ | Solid cylinder gripped in right arm |
| $G_5$ | Hollow cylinder on stick for stacking objects |
| $G_6$ | Solid cylinder in box |

Table 3
P-nodes learned after the execution of the experiment

| P-node | Description |
|---|---|
| $P_1$ | Box out of reach |
| $P_2$ | Hollow cylinder on the left side of the table and left arm free |
| $P_3$ | Hollow cylinder gripped in the left arm and right arm free |
| $P_4$ | Solid cylinder on the right side of the table and right arm free |
| $P_5$ | Hollow cylinder gripped in right arm |
| $P_6$ | Solid cylinder gripped in right arm |

the dreaming-like method explained in Section 4, the policies corresponding to the six UMs have also been learned. Tables 2–4 contain a summary of the content of these knowledge nodes as described by an external observer. As can be seen, the P-nodes correspond to the different perceptual situations that the robot may encounter, while the UMs and policies represent the actions to be performed to go from these perceptual situations to the associated goals. To provide an idea of the statistical distribution of the behavior of the learning process, Fig. 5 shows the results, in terms of purpose achievement, resulting from 10 different runs of the experiment. The experiments shown in Figs 4 and 5 are divided into trials, each of which ends when the robot achieves its purpose. Similarly, a trial is divided into time steps, where each time step represents the execution of an action. Remember that each action $A(t)$ moves the robot's arms a maximum of 5 centimeters in a particular direction, so that to reach a goal it is necessary to concatenate several actuations.

At a first glance, it can be seen from the figures that there is a gradual reduction in the number of time steps required to achieve the robot's purpose as the robot acquires more knowledge. Moreover, this reduction is associated with the learning of the different UMs and P-nodes. Something that was expected, since as the robot learns to reach the different goals, the time to achieve its purpose also decreases. Furthermore, once it learns how to achieve the last goal, the number of steps needed to achieve the purpose converges. A video illustrating the robot operation once all the UMs and P-nodes have been learned can be found in the following link: https://github.com/GII/ICAE2023.

Table 4
Utility models and policies learned after the execution of the experiment

| Utility model /policy | Description |
|---|---|
| $UM_1/\pi_1$ | Reduce the distance between the left arm and the button |
| $UM_2/\pi_2$ | Reduce the distance between the left arm and the hollow cylinder |
| $UM_3/\pi_3$ | Reduce the distance between the right arm and the hollow cylinder |
| $UM_4/\pi_4$ | Reduce the distance between the right arm and the solid cylinder |
| $UM_5/\pi_5$ | Reduce the distance between the right arm and the stick to stack objects |
| $UM_6/\pi_6$ | Reduce the distance between the right arm and the box |

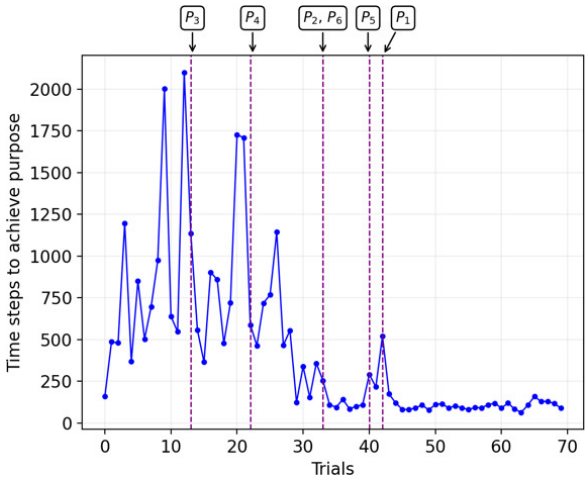


Fig. 4. Number of iterations required to achieve the purpose in a representative run of the experiment. Since the UMs and P-nodes related to the same goal are learned in parallel, the end of the learning process is the same for both.

### 5.3. *P-node learning*

The different P-nodes, as representations of perceptual classes, are constructed starting from the initial perception (state space point) that led the robot to reach each one of the goals for the first time. From that moment on, the P-node presents an activation value of 1.0 for that representative point and lower values (depending on the representation function used) in the hypothesis area surrounding it. This initial delimitation of the P-node is only a hypothesis and, consequently, some of the area it encompasses may actually not be part of the perceptual class.

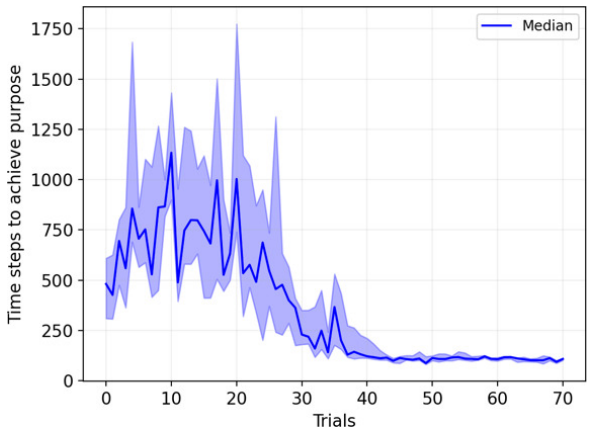


Fig. 5. Performance analysis for 10 runs of the experiment (median and 25 and 75 percentiles).

As interactions with the environment take place involving a P-node, more information will become available, i.e., more points (P-node activating perceptions from which it was possible to reach the related goal by using the related UM) and anti-points (P-node activating perceptions, but from which, in this case, it was not possible to reach the related goal). This will lead to a progressive improvement of the delimitation of hypothesis areas corresponding to the perceptual classes. Consequently, the robot will become more efficient when trying to achieve its goals.

To illustrate this process, Fig. 6 shows the evolution in time of the hypothesis area of P-node $P_1$. Given that in this experiment the perceptual space is eighteen-dimensional, to facilitate the visualization of the data the figures show the P-node activation map associated to the most relevant two dimensions. That is, the two sensors that play the most important role in the calculation of the activation have been chosen. In this case, as it is a P-node associated with pressing the button, and it is only useful to press it when the box is out of the robot's reach, the most significant sensors are those corresponding to where the box is (distance and angle between the robot and the box). Bear in mind that this is just for visualization purposes, as the robot has learnt the eighteen-dimensional P-node activation map on its own and determined when it was relevant for the P-node to be activated. With this representation we want to provide an idea of how the perceptual classes work and of their importance when determining in which regions of the state space to use the different UMs.

Figure 6 shows the areas of the state space in which the P-node will be activated and in which it will be inhibited. The darker zones (higher activation values) are those that correspond to perceptions that are included within the perceptual class. From the time evolution sequence of images in the figure, it can be appreciated that when the robot has not interacted much with the environment, the perceptual class boundaries are quite poor. In fact, in this particular case, the hypothesis made by the robot on the perceptual class clearly shows that in trial 10 the robot has not experience related to when the box is on the right side of the table. However, as more points and anti-points are experienced, a much more detailed delimitation of the P-node area is obtained, leading to more accurate decisions based on it.

Figure 7 shows a similar representation to that on Fig. 6 for some P-nodes at the end of the execution of the experiment. The top graph displays the joint activation areas of P-nodes $P_2$ and $P_4$, which are the ones related to grasping the different objects. Thus, in the figure it can be seen how when the objects (cylinders) are close, i.e., within the robot's reach area, the P-nodes are activated indicating that the robot can reach them. It is worth noting that this representation is complementary to that of the perceptual class shown in Fig. 6, which indicates when the object, in that case the box, is out of reach.

Another interesting plot is the one corresponding to P-node $P_3$ (Fig. 7 bottom), associated with the UM corresponding to moving an object from one gripper to the other when stacking objects. The most significant sensors are those that allow the robot to determine whether the object and the stacking stick are on the same side or not. In this case they are the angle from the robot base to the object (cylinder with an axial hole) and the angle from the robot base to the stick used to stack objects. To put the hollow cylinder there, the robot first needs to have this object in the gripper that is on the same side as the stick. This is what the perceptual class shows, when the hollow cylinder is not on the same side as the stick the robot must move it from its current gripper to the other one.

The main conclusion to be drawn from these figures is that P-nodes are a good tool for partitioning the perceptual space in terms of the context (domain and task) to be addressed. Furthermore, they are key to delimit the perceptual conditions necessary to try to achieve certain goals, as they create well-established boundaries that delimit where the P-node should be activated and, thus, where the UM that is being used to evaluate possible actions is valid. This is a fundamental issue, as we will comment in the next section, when trying to learn policies from deliberative processes that involve these learnt UMs. In addition, it must be noted that, as appreciated in Fig. 4, the correct delimitation of these

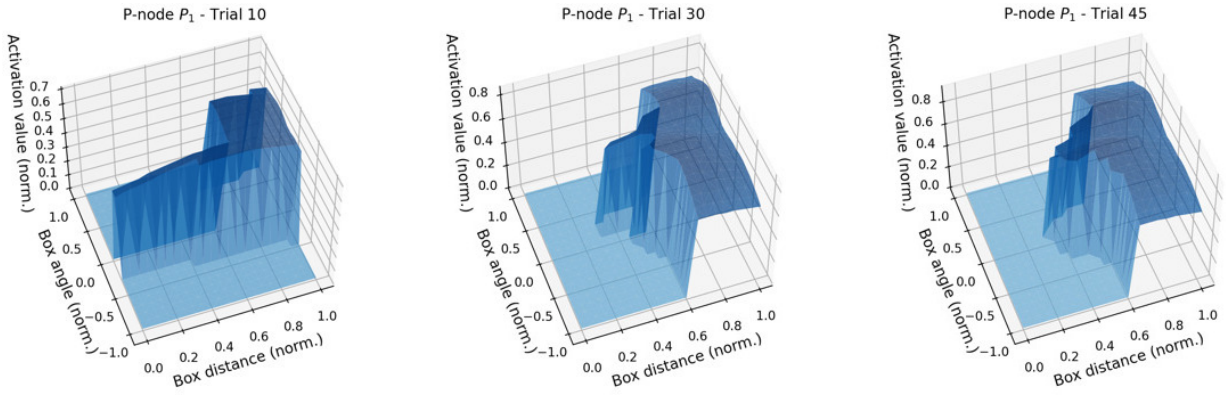


Fig. 6. Evolution of the activation areas of P-node $P_1$ for trials 10, 30 and 45.

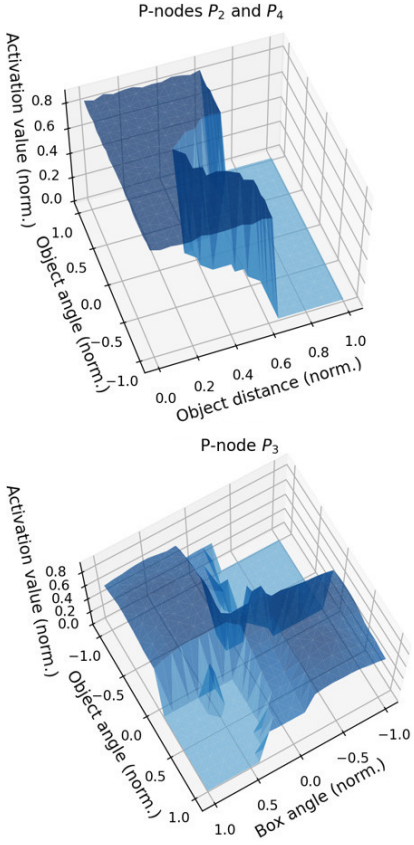


Fig. 7. Final activation areas for P-nodes $P_2$ and $P_4$, and $P_3$.

classes is linked to an increase in the performance of the system when achieving its purpose as the system makes fewer erroneous evaluations of actions.

### 5.4. Policy learning and results

As the learning process was the same for each policy, for the sake of brevity we will explain it for the ones corresponding to depositing the hollow cylinder on the base to stack objects, $UM_5$, and to moving the cylinder from one arm to the other, $UM_3$. For clarity, they will be referred to in what follows as "*Put object in*" and "*Change hands*". These policies were chosen because they present different complexity levels. In the case of the first one, $\pi_5$ or "*Put object in*", given that objects were always presented on the right side of the robot, it "only" has to control the movement of the right arm. In other words $\pi_5 = f(\alpha_{left}, v_{left})$. In the case of the second one, $\pi_3$ or "*Change hands*", it must control the movement of both the right and left arm, $\pi_3 = f(\alpha_{left}, v_{left}, \alpha_{right}, v_{right})$.

As mentioned in previous sections, to evaluate each of the candidate policies generated in the evolutionary process, the previously learned UMs were used. They will provide a higher utility value (higher fitness) to those actions that satisfy the description shown in Table 4 for models $UM_3$ and $UM_5$. Thus, in the case of "*Put object in*" candidate policies, those that provide actions that allow the robot's right arm to move closer to the stick for stacking objects will be valued higher. While in the case of "*Change hands*" policies, the best valued ones will be those that provide actions that allow the robot arms to get closer to each other.

Additionally, to be able to evaluate candidate policies, it is essential to know the areas of the state space in which the UMs that are used for this are valid. This is where the perceptual classes play a key role since they provide the necessary information to "contextualize" the learning. It is worth remembering that in LOLA conditions this information is not available beforehand, and that P-nodes are learned autonomously. Thus, during the dream-like learning process, these perceptual classes are used in order to generate points (dreamt perceptions) belonging to the area of the perceptual state

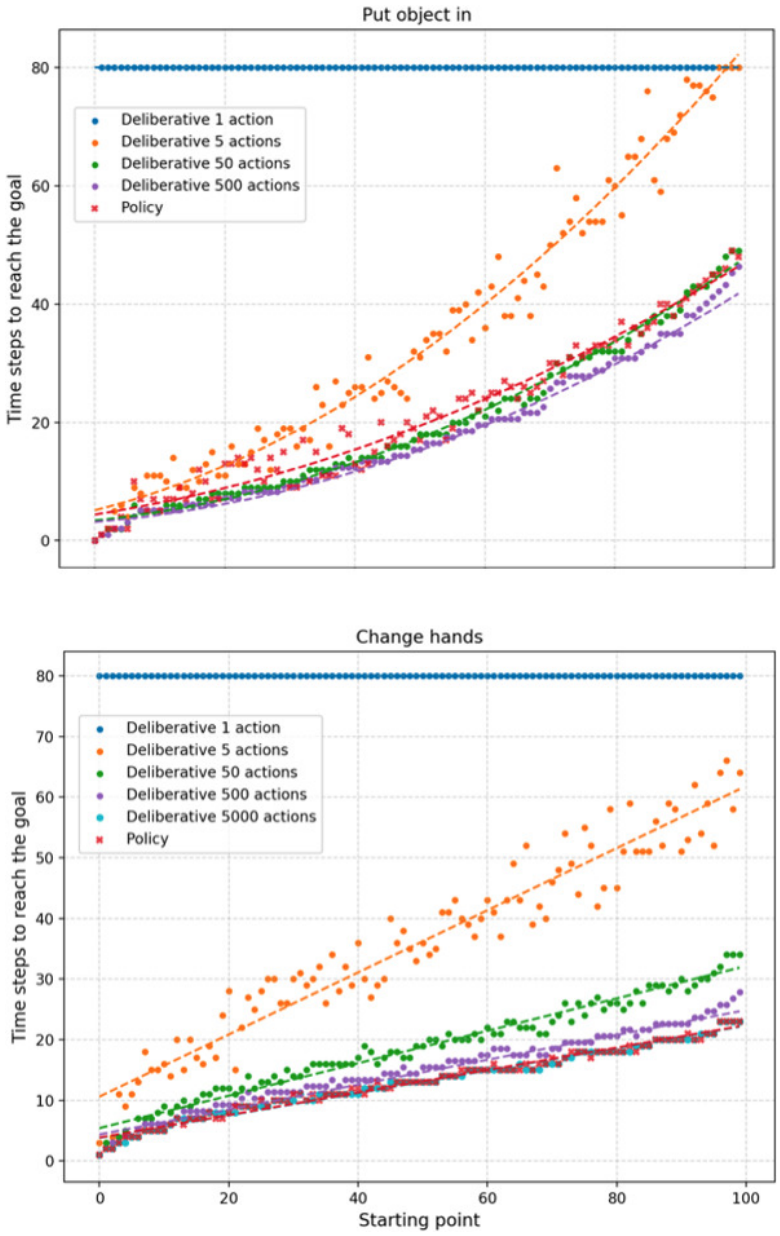


Fig. 8. Time steps needed to achieve the goals using the learned policies and deliberative models considering different numbers of actions. A maximum of 80 time-steps were allowed to reach the goal.

space in which the policies are going to be used, that is, where the UMs of the deliberative processes employed during dreaming are valid. These perceptions, $p_k$, will be the inputs of the candidate policies and will allow obtaining the actions that will then be evaluated with the help of the world and UMs in a deliberative process. To illustrate this stage of the autonomous learning process, Fig. 7 bottom shows a representation of the state space covered by the P-node associated to goal $G_3$.

To evaluate the efficiency of the policies obtained after the evolutionary process, we have compared the actions they provide with those that would be provided by the deliberative process using the learnt UMs. Figure 8 shows a representation of the number of time steps (action selections) needed to reach the goals using the learned policies. To this end, 100 random starting points from within the P-nodes associated to these UMs/policies are shown. These starting points are ar-

ranged according to their increasing distance to the goal, so that the further to the right the starting point is, the farther the robot will be from reaching the goal and the more actions it will have to take in order to achieve it.

In turn, to draw conclusions regarding the efficiency of the actions obtained using the policy, the number of time steps necessary to reach the goal is compared to those that would be necessary if the deliberative system considered 1, 5, 50, 500 and 5000 candidate actions during its deliberations. It should be remembered that these actions are generated at each time instant, that they take random values in the continuous range of possible actions and that they control the direction of movement and the displacement of the robot grippers. Therefore, the larger the repertoire of possible actions, the greater the probability of finding the optimal action for each situation. However, this will also imply a longer computational time since more actions will need to be evaluated (each one of them will imply prospection and evaluation). In the case of the policy, it directly provides the action it considers optimal, without the need for evaluation.

Figure 8 top shows the results corresponding to policy '*Put object in*'. It should be remembered that each action involves moving the robot arm a maximum of 5 centimeters in a particular direction, so that to reach an object several actions are necessary. Thus, in the figure it can be seen how the actions provided by the policy are very close to being optimal, since, on average, it needs only two more actions than the best deliberative case to reach the goal. This difference arises because the robot needs high precision in the final step towards the goal. That is, if the last actions imply too large a displacement, the robot will overshoot the position of the base to drop the object. Thus, to prevent this from happening, the policy chooses to use shorter displacements as the robot approaches the goal, which results in a slight loss of efficiency.

On the other hand, observing the results obtained for the 'Change hands' policy (Fig. 8 bottom), it is possible to verify that these are optimal. The actions proposed by the policy are as good as those of the deliberative system that contemplates 5000 candidate actions. It is interesting to comment that the deliberative systems need more time steps to reach this goal than the goal associated with policy "Put object in". This is because these actions control the two arms of the robot, so that each candidate action (randomly generated) has four components, $(\alpha_{left}, v_{left}, \alpha_{right}, v_{right})$, enlarging the candidate action search space. Thus, the more complex the actions that are applied to control the robot, the more difficult it will be to find the optimal action through deliberation, and the more valuable it will become to have a policy that can do it.

To end the comparison, it is worth mentioning that the use of policies is much less computationally expensive. The reason is clear: each candidate action in a deliberative process involves the evaluation of two ANNs, one corresponding to the WM and one corresponding to the UM. Thus, $2n$ evaluations will be needed, being $n$ the number of candidate actions. Whereas in the case of policies, only one ANN needs to be evaluated. To demonstrate this, we have used an intel i7-7700HQ @ 2.80 GHz CPU and computed the average time for obtaining 100 different actions with each of the systems shown in Fig. 8. The order of magnitude of the computational time for the policy was $10^{-5}$ seconds, the same as for the deliberative process with 1 candidate action. In the case of 5 candidate actions, the time was $10^{-4}$ seconds, while for 50 candidate actions it was $10^{-3}$ seconds. In the same way, the average time for 500 actions was $10^{-2}$ seconds and for 5000 actions was $10^{-1}$ seconds.

To conclude, it was possible to see that the autonomous learning of policies through dreaming processes in LOLA conditions is feasible. This is possible by using the knowledge contained in the world and UMs and, above all, the information contained in the perceptual classes that provide an indication of where these models are valid and thus allow the generation of relevant starting state space points in order to generate dreamt trajectories that permit evaluating actions. It is thus possible to contextualize learning, without the need to know in advance the goals and the domains of operation.

## 6. Conclusions

This paper has proposed and tested a new approach towards online policy learning in the realm of autonomous robotics. The approach is different from others existing in the Reinforcement Learning literature, mainly, because it was developed to provide robots with Lifelong Open-ended Autonomy (LOLA). That is, with the capability of reaching their design objectives in unknown and intermingled domains, in periods of time with no predefined ending, and in real conditions of operation. As it can be supposed, online policy learning is really challenging under these conditions.

The current work has shown, through a real robot experiment in LOLA settings, a possible path towards

addressing this issue. It is based on a motivational system which relies on the domain-independent concepts of need and drive, which guide the lifelong operation of the robot. With a proper combination of cognitive and operational drives, goal discovery is possible, even with intermingled domains, as shown in the experiment. In addition, a learning method to determine expected utility under LOLA constraints was also tested, so that continuous utility models have been inferred online.

But the main contribution of this work is focused on the online policy learning process itself. Deliberative policy learning has been addressed in the realm of Reinforcement Learning with success. However, LOLA conditions imply that it cannot be assumed that the learned world and utility models are valid throughout the complete state space of the robot for a given domain as in the case of RL. Hence, state inputs that are relevant must be provided to the dreaming deliberative process to generate data that allows learning a policy. In this line, we have implemented and tested a policy learning process that involves Perceptual Classes as a metacognitive structure in the form of P-nodes. They are contextually linked to goals, world models, and utility models, and they are also learned on-line. During normal operation, they allow inferring the relevant perceptual domain from which the robot is able to achieve the goal and thus for which a given expected utility model is valid. Consequently, they can be used to generate feasible data points that permit generating trajectories for deliberative policy learning in a very efficient way. Perceptual Class learning and application has also been successfully tested in a real-life experiment.

From the results obtained in this paper, promising future work arises in many aspects, with the aim of solving more realistic tasks in real robot conditions. One of the most challenging topics for the near future is related with the integration of State Representation Learning strategies in LOLA, which has been started with a preliminary work in [36]. It is necessary to study the management of multiple representations within a cognitive architecture, so that it is possible to chain multiple representation functions that gradually reduce the dimensionality of the state space and facilitate learning and decision-making. In this line, it would also be valuable to study in more detail the introduction of attention mechanisms and their integration with the representation system, so that it is possible to achieve task or goal-oriented representations. Finally, newer supervised machine learning/classification algorithms will be tested to improve P-node learning and model learning such as Neural Dynamic Classification algorithm [47], Dynamic Ensemble Learning Algorithm [48], and variations of the NEAT algorithm used here [49].

## Acknowledgments

This work was partially funded by MCIN/AEI/10.13039/501100011033 (grant PID2021-126220OB-I00) and by "ERDF A way of making Europe", Xunta de Galicia (grant EDC431C-2021/39), Centro de Investigación de Galicia "CITIC" (grant ED431G 2019/01), and by Horizon Europe, GA 101070381 'PILLAR-Robots – Purposeful Intrinsically-motivated Lifelong Learning Autonomous Robots'.

## References

[1] Hernandez-Barragan J, Lopez-Franco C, Arana-Daniel N, Alanis AY, Lopez-Franco A. A modified firefly algorithm for the inverse kinematics solutions of robotic manipulators. Integr Comput Aided Eng. 2021; 28(3): 257-75.

[2] Schwan C, Schenck W. A three-step model for the detection of stable grasp points with machine learning. Integr Comput Aided Eng. 2021; 28(4): 349-67.

[3] Doncieux S, Filliat D, Díaz-Rodríguez N, Hospedales T, Duro R, Coninx A, et al. Open-ended learning: A conceptual framework based on representational redescription. Front Neurorobot. 2018; 12(SEP): 1-6.

[4] Thrun S, Mitchell TM. Lifelong robot learning. Rob Auton Syst. 1995 Jul 1; 15(1-2): 25-46.

[5] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press Cambridge; 1998. vol. 1.

[6] Zahra O, Navarro-Alarcon D, Tolu S. A neurorobotic embodiment for exploring the dynamical interactions of a spiking cerebellar model and a robot arm during vision-based manipulation tasks. Int J Neural Syst. 2022; 32(8): 2150028.

[7] Macias-Garcia E, Galeana-Perez D, Medrano-Hermosillo J, Bayro-Corrochano E. Multi-stage deep learning perception system for mobile robots. Integr Comput Aided Eng. 2021; 28(2): 191-205.

[8] Gil-Gala FJ, Mencía C, Sierra MR, Varela R. Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. Integr Comput Aided Eng. 2021; 28(1): 65-80.

[9] Gasienica-Jozkowy J, Knapik M, Cyganek B. An ensemble deep learning method with optimized weights for drone-based water rescue and surveillance. Integr Comput Aided Eng. 2021; 28(3): 221-35.

[10] Avola D, Cascio M, Cinque L, Foresti GL, Pannone D. Machine learning for video event recognition. Integr Comput Aided Eng. 2021; 28(3): 309-32.

[11] Liu H, Gu F, Lin Z. Auto-sharing parameters for transfer learning based on multi-objective optimization. Integr Comput Aided Eng. 2021; 28(3): 295-307.

[12] Guzman C, Castejon P, Onaindia E, Frank J. Reactive execution for solving plan failures in planning control applications. Integr Comput Aided Eng. 2015; 22: 343-60.

[13] Baldassarre G, Mirolli M. Intrinsically motivated learning systems: an overview. Intrinsically Motiv Learn Nat Artif Syst. 2013; 1-14.

[14] Santucci VG, Oudeyer PY, Barto A, Baldassarre G. Intrinsically motivated open-ended learning in autonomous robots. Front Neurorobot. 2020; 115.

[15] Yu Y, Chang AYC, Kanai R. Boredom-driven curious learning by homeo-heterostatic value gradients. Front Neurorobot. 2019; 88.
[16] Schmidhuber J. A possibility for implementing curiosity and boredom in model-building neural controllers. In: Proc of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats. 1991. pp. 222-7.
[17] Schmidhuber J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE Trans Auton Ment Dev. 2010; 2(3): 230-47.
[18] Oudeyer PY, Kaplan F, Hafner VV. Intrinsic motivation systems for autonomous mental development. IEEE Trans Evol Comput. 2007; 11(2): 265-86.
[19] Hester T, Stone P. Intrinsically motivated model learning for developing curious robots. Artif Intell. 2017; 247: 170-86.
[20] Mannella F, Mirolli M, Baldassarre G. Goal-directed behavior and instrumental devaluation: a neural system-level computational model. Front Behav Neurosci. 2016; 10: 181.
[21] Huang X, Weng J. Value system development for a robot. In: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat No 04CH37541). 2004. pp. 2883-8.
[22] Merrick K. Value systems for developmental cognitive robotics: A survey. Cogn Syst Res. 2017; 41: 38-55.
[23] Prieto A, Romero A, Bellas F, Salgado R, Duro RJ. Introducing separable utility regions in a motivational engine for cognitive developmental robotics. Integr Comput Aided Eng. 2018; 26(1).
[24] Romero A, Bellas F, Prieto A, Duro RJ. Utility Model Re-description within a Motivational System for Cognitive Robotics. In: IEEE International Conference on Intelligent Robots and Systems. 2018.
[25] Zhao G, Tao Y, Liu H, Deng X, Chen Y, Xiong H, et al. A robot demonstration method based on LWR and Q-learning algorithm. J Intell Fuzzy Syst. 2018; 35: 35-46.
[26] Peters J, Schaal S. Policy gradient methods for robotics. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2006. pp. 2219-25.
[27] Grondman I, Busoniu L, Lopes GAD, Babuška R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Trans Syst Man Cybern Part C Appl Rev. 2012; 42(6): 1291-307.
[28] Deisenroth MP. A Survey on Policy Search for Robotics. Found Trends Robot. 2011; 2(1-2): 1-142.
[29] Packer C, Abbeel P, Gonzalez JE. Hindsight task relabelling: Experience replay for sparse reward meta-rl. Adv Neural Inf Process Syst. 2021; 34: 2466-77.
[30] Lillicrap T, Ba J, Wu A, Ryoo MS. Learning Real-World Robot Policies by Dreaming. 2020. pp. 1-20.
[31] Lillicrap T, Ba J. Dream to Control: Learning Behaviors by Latent Imagination. 2020. pp. 1-20.
[32] Bellas F, Duro RJ, Faiña A, Souto D. Multilevel darwinist brain (MDB): Artificial evolution in a cognitive architecture for real robots. IEEE Trans Auton Ment Dev. 2010; 2(4): 340-54.
[33] Romero A, Bellas F, Becerra JA, Duro RJ. Motivation as a tool for designing lifelong learning robots. Integr Comput Aided Eng. 2020; 27(4).
[34] Piergiovanni A, Wu A, Ryoo MS. Learning Real-World Robot Policies by Dreaming. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019. pp. 7680-7.
[35] Romero A, Piater J, Bellas F, Duro RJ. ANN-based Representation Learning in a Lifelong Open-ended Learning Cognitive Architecture. In: 2022 International Joint Conference on Neural Networks (IJCNN). 2022 July.
[36] Romero A, Meden B, Bellas F, Duro RJ. Autonomous Knowledge Representation for Efficient Skill Learning in Cognitive Robots. In: Bio-Inspired Systems and Applications: From Robotics to Ambient Intelligence: Proceeding of the 9th International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2022, Puerto de La Cruz, Tenerife, Spain, May 31–June 3, 2022. Berlin, Heidelberg: Springer-Verlag; 2022. pp. 253-263. doi: 10.1007/978-3-031-06527-9_25.
[37] Hawes N. A survey of motivation frameworks for intelligent systems. Artif Intell. 2011; 175(5-6): 1020-36. doi: 10.1016/j.artint.2011.02.002.
[38] Huang X, Weng J. Novelty and reinforcement learning in the value system of developmental robots. In: Proceedings of the 2nd International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. 2002. p. 55.
[39] Colas C, Fournier P, Chetouani M, Sigaud O, Oudeyer PY. Curious: intrinsically motivated modular multi-goal reinforcement learning. In: International Conference on Machine Learning. 2019. pp. 1331-40.
[40] Oudeyer PY, Kaplan F, Hafner VV, Whyte A. The playground experiment: Task-independent development of a curious robot. In: Proceedings of the AAAI Spring Symposium on Developmental Robotics. 2005. pp. 42-7.
[41] Santucci VG, Baldassarre G, Mirolli M. Which is the best intrinsic motivation signal for learning multiple skills? Front Neurorobot. 2013; 7: 22.
[42] Romero A, Bellas F, Becerra JA, Duro RJ. Bootstrapping Autonomous Skill Learning in the MDB Cognitive Architecture. LNCS, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2019. vol. 11486.
[43] Duro RJ, Becerra JA, Monroy J, Bellas F. Perceptual Generalization and Context in a Network Memory Inspired Long-Term Memory for Artificial Cognition. Int J Neural Syst. 2019; 29(6): 1-26.
[44] Becerra JA, Duro RJ, Monroy J. A Redescriptive Approach to Autonomous Perceptual Classification in Robotic Cognitive Architectures. In: Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN). 2018 July. p. 640891.
[45] Stanley KO, Miikkulainen R. Evolving Neural Networks through Augmenting Topologies. Available from: http://direct.mit.edu/evco/article-pdf/10/2/99/1493254/1063656023 20169811.pdf.
[46] GitHub. MultiNEAT/MultiNEAT: Portable NeuroEvolution Library http://MultiNEAT.com. Available from: https://github.com/MultiNEAT/MultiNEAT.
[47] Rafiei MH, Adeli H. A new neural dynamic classification algorithm. IEEE Trans Neural Networks Learn Syst. 2017; 28(12): 3074-83.
[48] Alam KMR, Siddique N, Adeli H. A dynamic ensemble learning algorithm for neural networks. Neural Comput Appl. 2020; 32: 8675-90.
[49] Papavasileiou E, Cornelis J, Jansen B. A systematic literature review of the successors of "neuroevolution of augmenting topologies". Evol Comput. 2021; 29(1): 1-73.