Developmental Learning of Value Functions in a Motivational System for Cognitive Robotics

Alejandro Romero
GII, CITIC research center
University of A Coruña
A Coruña, Spain
alejandro.romero.montero@udc.es

Francisco Bellas
GII, CITIC research center
University of A Coruña
A Coruña, Spain
francisco.bellas@udc.es

Abraham Prieto
GII, CITIC research center
University of A Coruña
A Coruña, Spain
abprieto@udc.es

Richard J. Duro
GII, CITIC research center
University of A Coruña
A Coruña, Spain
richard@udc.es

***Abstract*— Motivation is quite an important topic when addressing continual open-ended learning processes in autonomous robots. The three main issues that need to be considered are, firstly, how does a designer define what the robot strives for in a manner that is independent from any particular domain it may find itself in. Secondly, once that robot is in a domain, how does it go about finding and relating goals in that particular domain on its own. Finally, the third issue is, once a goal is found, how does a robot establish a representation, usually in the form of a Value Function, that will allow it to exploit that goal. This paper deals with the third issue in the framework of the motivational engine we have designed for cognitive architectures. It addresses the problem of efficiently and appropriately learning complex Value Functions starting from intrinsically motivated traces of valuated robot actions that are often ambiguous and multivalued. To this end, a developmental learning mechanism is proposed that relies on the concurrent application of a real time ANN learning procedure over the traces of the valuated robot actions, and a simpler sensor correlation-based approach to allow for the production of better configured data traces for the learning process. The mechanism is analyzed and discussed over an experiment considering a real Baxter robot.**

Keywords—motivation, cognitive developmental robotics, open-ended learning, value function learning

I. INTRODUCTION

Barto [1] described motivation as a series of "processes that influence the arousal, strength and direction of behavior". Any autonomous robot that must operate in open-ended learning settings [2], must be endowed with some type of motivational mechanism that defines what it should strive for at any given point in time [3]. This motivational mechanism is the one that structures the evaluation required by any decision process the robot cognitive architecture must perform, and it does so by determining at each point in time the goals the robot must achieve. Thus, when deciding among different alternatives, the robot evaluates each one of them by estimating how useful they may be for achieving its current goal, and chooses the one that is most useful, that is, the one that provides the highest utility with regards to that goal. It is through this evaluation process that motivation is reflected in the actions the robot performs.

Looking at this more formally from a deliberative point of view, if we have a robot in its current state $S_t \epsilon \mathcal{S}$ (being $\mathcal{S}$ the robot's state space) at time t, it will always try to choose the best action to perform, ${}^{*}A_t$ (where * denotes the optimum) by, first, prospectively exploring the consequences of a number of different possible actions $\{A_1, A_2, \ldots A_n\}$ from its action repertoire that could be applied at time t. These consequences can be expressed in terms of the set of end-states achieved by the actions, ${}^{i}S_{t+1}$ with $i\epsilon[1,n]$ and, consequently, related to them. Thus, by determining the utility of these end-states towards the goal, the one with the highest utility, ${}^{*}S_{t+1}$, can be chosen, leading to the determination of the best action to be applied (i.e. the one that led to the optimum prospective state). Of course, performing prospection requires the availability of a *World Model (WM)*, and to determine the utility of states, a *Utility Model (UM)* is necessary [4].

In the realm of reinforcement learning (RL) [5], the functions used for determining utility are called *Value Functions (VF)*, and they have been studied extensively [6][7]. They represent the expected utility of each point in state space with regards to a goal. That is, the probability of achieving a utility starting from that point. However, in most of the work in RL, it is usually assumed that the robot will operate in a known domain, where the robot designer is able to predefine the goal or goals that need to be achieved in terms of points in the state space of the robot in that domain it needs to reach (e.g. find red balls). In other words, the designer is able to express what the robot needs to do (its task) in terms of an operational goal defined as the specific perceptions the robot needs to achieve in that particular domain.

In open-ended learning settings, however, this is not possible. That is, there is no way for the designer to define these goals beforehand as, by definition of open-endedness, the domains the robot will operate in are unknown at design time. Consequently, in order to be able to produce meaningful oriented behavior in the robots, different approaches in the literature [8][9] and, in particular, the motivational engine we have defined [4], have resorted to the definition of a motivational structure based on *drives*. Drives can be seen as the distance to internal goals of the robot that are completely independent from the domain it is in. These goals are defined in an internal (sometimes called motivational) state space, which is not directly related to the operational state space ($\mathcal{S}$) defined by the combination of the robot and a particular domain. Drives are established by the system designer, and the value of each drive reflects how far the system is from satisfying a need the designer

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

has deemed important for the robot, independently of how this need can be satisfied in any particular domain.

Consequently, in a motivational architecture for open-ended operation, a designer defines the needs and associated drives of the robot. When the robot finds itself in a new domain it needs to find goals (state space points, S_t, that produce utility) in that particular domain and learn how to exploit them in order to be able to satisfy its drives. Unlike in traditional RL, the robot is establishing its own goals in the domain and in the process of finding ways to exploit them, it is learning new policies or skills. It is important to note that the only control over a cognitive robot a designer has is by creating the appropriate drives.

It is often the case that to be able to find a goal that actually fulfills an operational drive it is necessary to first acquire some basic policies or skills (e.g. to get an apple from a tree, we need to learn to reach the tree and to climb the tree before being in a position to learn that an apple satisfies our hunger). Thus, if the designer wants the robot to explore and acquire those skills, it needs to provide the robot with drives that will allow establishing some type of indirect or virtual goal that permits performing these learning processes. We classify these drives as cognitive drives, that is, drives that are there to induce learning and whose satisfaction is related to how much is explored as in the case of novelty [10], to how much is learnt as in the case of curiosity [11][12] or to how much effect is achieved over the environment as in the case of effectance [13]. Cognitive drives have often been assimilated to *Intrinsic Motivations (IMs)*, following the nomenclature established in the psychological and educational literature [14], [15]. This term was defined by Ryan and Deci [16] as "the doing of an activity for its inherent satisfaction rather than for some separable consequence".

Many authors have resorted to intrinsic or cognitive drives to try to directly find goals or to pre-learn skills that can facilitate finding goals [16]. In fact, in a previous paper [18], we have described a developmental mechanism that allows for the progressive construction of a complete drive-goal structure in open-ended settings starting from a combination of cognitive and operational drives.

However, if, once an operational goal is found, a robot is to be able to consistently choose actions leading to that goal in a deliberative manner, it is necessary for the robot to construct a Value Function that is associated to it. This problem is a hard problem due to the complexity of the domains the robot may find itself in and to the fact that they are not known beforehand, implying that these models need to be obtained on-line, but, more importantly, due to the fact that the only information the robot has are the traces of states it went through as it applied actions in that domain until the goal was reached.

The most common approach to the construction of VFs when in continuous domains that are not known beforehand is the use of an eligibility traces based strategy [4]. The idea is to provide an expected utility value to each step in a trace that was successful in reaching a goal that decays from the goal with the position of the step on the trace, and use the values obtained from a large enough set of traces in order to define or learn the final VF. Although popular, this approach presents many problems when considering real, continuous and complex domains due to the inconsistencies, ambiguities and multi-valuations of points that may arise, which can lead to very poor VF modelling results using these techniques.

A. Scientific contribution

In this paper, we face the problem of the construction through learning of Artificial Neural Network (ANN) based Value Functions in open-ended learning scenarios. In particular, we address the issue of the very poor conformation of the training sets obtained directly from cognitive drive induced traces in complex domains. To this end, we propose a developmental learning approach that can be applied in any domain. This approach starts by creating very rough local Utility Models based on the concept of Separable Utility Regions (SURs) [19], which are used to conform well behaved traces, without ambiguities, that can be used as training data in a subsequent ANN based VF learning stage. We show that the results produced through this combined approach improves on the traditional ANN based VF learning procedures.

II. Description of the Problem

A. General Value Function Learning

To learn Value Functions from traces using ANNs is a typical approach in RL. It is the approach we have used in previous works within the motivational system of the Epistemic Multilevel Darwinist Brain cognitive architecture (e-MDB) [4], as explained in detail in [19] and [20]. Specifically, our approach was based on the use of cognitive drives to generate exploratory behaviors in order to locate goals. Then, when a goal had been found, we made use of the information on the trace of states followed to reach the goal as an initial training set to train an ANN as a VF for that goal. Obviously, this small data set of only one trace will not produce a reliable VF and a balance between following the VF in order to reach the goal and use the cognitive drive to explore other paths towards the goal that can add new training data in order to improve the VF had to be established. Once a large enough set of traces with well evaluated states was gathered, the training of the VF should be optimal. In what follows we describe this process in more detail organized into 3 stages:

1) Initial steps

In the initial stages of learning, no goal state has been reached, and as there are not VFs available for reaching goals corresponding to operational drives, the motivational system makes use of a cognitive drive (C-Drive), which guides the robot behavior towards the discovery of unvisited sensorial states operating as an explorative intrinsic process. One possibility is to use the concept of novelty when choosing what states to spend efforts on reaching out of a set of candidate states that are prospectively generated. Novelty is specified as a distance measurement between perceptual states that tries to steer the robot towards areas of its state space that are most unknown. To compute it, a trajectory buffer is created that stores all the perceptual states the robot has experienced in the last M instants of time. Formally, the novelty of the *k-th* candidate state $S_{c,k}$ is:

$$Nov_k = \frac{1}{M}\sum_{i=1}^{M} dist(S_{c,k} - S_i)^n \quad with\ k = 1\ to\ N$$

where n is a coefficient that regulates the balance between the relevance of distant and near states, S_i is the i-th state in the trajectory buffer and N is the number of candidate states considered. High values of n guide the exploration towards distant states while low values of n lead to a more local exploration.

A novelty value is assigned to each of the N candidate states proposed, and the one with a highest value will be chosen as the desired state. As a consequence, the cognitive architecture will apply the action or actions required to reach such state [20].

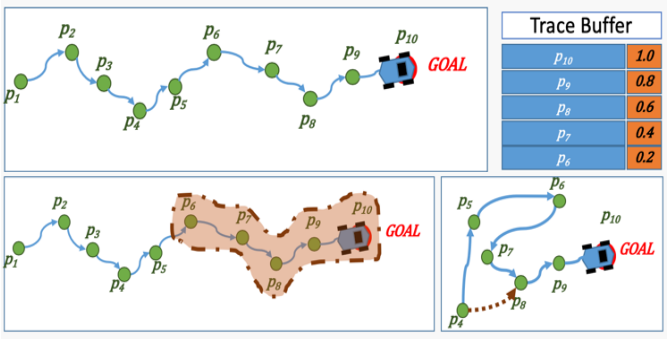

Fig. 1. Schematic representation of the VF learning process.

2) First goal achieved

Fig. 1 helps to illustrate this second stage. The first time a robot reaches a goal using the C-Drive, it receives a real utility value (top left image of Fig. 1). Following the eligibility traces strategy of RL [5], a first trace of real data, made up of the series of state space points the robot went through as it reached the goal, can be stored in the Trace Buffer. This is represented on the top right diagram of Fig. 1 with the points in the state space denoted by p_i (perceptions), and the assigned utility going from 1 at the goal point and decreasing to 0 along the trace, as usual in this type of strategy. Using these data as training set, an on-line VF learning process can start, as we will explain in detail in section 3. It is clear that modeling a VF using a single trace will provide a poor model, especially in complex scenarios. Therefore, the robot will need to reach the goal from different areas of state space in order to produce more traces that will make the training set more complete. That is, as more traces are obtained, the quality of the VF will increase.

3) Normal operation

Typically, in cognitive robotics, once the goal is reached, the setup is restarted in some way and the goal must be found again. This is a normal behavior of cognition-based systems, where the goal must be achieved many times to establish its real relevance. As a consequence, new data can be collected in order to improve the reliability of the VF. In the first steps, there are only a small number of traces in memory, so the reliability of the VF remains low. Also, in regions that are far away from those where the goal was found, the VF is not reliable either. Hence, a measurement that indicates the validity of the VF in those points is required.

To this end, we have developed the concept of certainty region, which represents the area in the state space where the VF is reliable and can be used to evaluate a candidate state. In [19], details on how the certainty regions are computed are provided, although the background idea is to calculate an n-dimensional area that encompasses the samples stored in the Trace Buffer. Continuing with the example of Fig. 1, the bottom left diagram shows a representation of a certainty region for this case, which surrounds the trace points that are stored in the Trace Buffer of the top right diagram, those where the VF is more reliable.

Consequently, during normal operation a candidate state will be evaluated using a given VF if it falls within the VF's certainty region. By construction, increasing values of a VF lead towards a goal. This implies that, whenever the robot finds its way into the certainty region of an appropriately defined VF, its deliberative mechanism will lead it up its slopes towards the goal. When outside any certainty region, the motivational mechanism will use a C-Drive, much in the same way as in the initial steps, with the objective of finding goals, that is, utility generating regions. Consequently, the appropriate modeling of VFs and the definition of their certainty regions are of paramount importance to be able to consistently achieve utility.

Obviously, as more traces are obtained, the reliability of the VF and the size of its associated certainty region will increase, thus making it more probable for a state to be encompassed by it. Consequently, the use of novelty will decrease, leading to an ideal situation where the relevant state space is covered by VFs, and the robot is able to reach any goal without having to search, just following the corresponding VFs.

B. Multivalued traces

It can be easily seen that this process has the problem of using an initial exploratory method to reach the goal, which could lead to a highly inefficient random search. To solve this issue, many techniques in the field of intrinsic motivation research have been proposed [1][3]. These try to find relevant directions in the state space that provide clues towards reaching the goal, being the novelty concept presented above one of them. Although these methods avoid a random search in the state space, some kind of exploration is required associated to any C-Drive. Hence, whenever a C-Drive is activated during the normal operation of the robot, non-optimal paths to the goal in the state space may arise, and these will lead to ambiguities and multi-valuations in the states stored in the traces.

This situation is illustrated in the bottom right representation of Fig. 1. Let us imagine that the robot has followed the path p_4, p_5, p_6, ...p_{10} to reach the goal. The resulting trace could be that of the top right table, where some states like p_7 will present a utility value that is higher than p_6, which would not be correct in terms of distance to the goal. In addition, a different trace could arise where the robot would go from p_7 directly to p_9. This would entail that p_7 would have two different expected utility values depending on the trace. Thus, in any learning process involving the traces of states as training set, learning will be severely hindered as there are points that are multivalued. Obviously, the optimal path would be moving from p_4 to p_8, but using C-Drives will often produce these types of ambiguities.

Hence, in the first stages of learning, while the C-Drive is applied, the states in the traces that are stored in the Trace Buffer may be multivalued, and any machine learning approach will have difficulties in modeling them. In previous research, we have been successfully using Deep Learning techniques over this Trace Buffer in relatively simple cases. But as the realism

and complexity of the problem increases, and the state space dimensionality becomes higher, proper learning becomes less accessible. To solve this issue and allow for the general learning of VFs in open-ended settings, we have designed a developmental learning methodology, which will be presented in the following section.

III. DEVELOPMENTAL LEARNING OF VALUE FUNCTIONS

The methodology proposed here, relies on a three-stage procedure in order to produce well behaved training sets that allow for robust ANN based Value Function learning. These three stages are in fact run concurrently and it is through their interactions that the desired result is achieved.

The first stage is the cognitive drive led exploration stage. This stage is the same as in the classical VF learning approach and is described in the previous section. It allows the robot to find goals and provides the initial traces that make up the training set. Here is where the approach diverges from the classical one. As these data may be poorly configured and present multi-valuations, and thus ambiguities, instead of directly using these traces in order to train the ANN corresponding to the global VF we will use them to produce a simplified Utility Model based on the concept of Separable Utility Regions (SUR) [19].

The idea behind a SUR is that the process of reaching a goal can be described by chaining sequences of actions in state space, each action chosen in order to increase or decrease a specific sensor value (implying a sort of Manhattan like motion, following one sensor each moment in time). In other words, the objective is to provide paths towards goals through sequences of direct correlations of utility value variations with the variation of the values of individual sensors. For instance, when the goal is to reach a light, the value of the light sensor is positively correlated with expected utility. Consequently, on the one hand, the multidimensional problem is divided into one-dimensional portions that are a lot easier to model. On the other hand, SURs do not represent expected utility values, but rather, an indication of in what directions these values grow (whether there a positive or a negative correlation). Obviously, this leads to a much rougher representation of utility, especially in complex situations. However, not having to produce exact expected utility values makes them very resilient to problems related to multi-valuation, randomness, ambiguities and to undetectable changes in the scenarios.

It is evident that the key to this approach is to determine the sensor tendencies as correlated to directions in state space that produce utility variations each moment in time (see Algorithm 1), choose the strongest one (the active sensor tendency) and define a certainty area for this correlation (for what area of state space this *sensor_value-utility_value_variation* correlation is valid). The main effort when learning a SUR comes from the definition of the corresponding certainty area, this is, the domain where the correlation is applicable. The computation of SURs and their certainty areas is explained in detail in [19] as well as how they work as utility or Value Function modelers by themselves. The main conclusion that is derived is that SURs are generally very easy to obtain even when the trace data is multivalued and otherwise ambiguous. Nonetheless, they do not represent a viable option in order to produce efficient and precise VFs, especially in complex domains.

The objective of this paper is not to produce final VFs made up of SURs, but rather, to make use of the resilience to poorly configured traces that SURs have demonstrated. Thus, in this stage of our procedure, the traces obtained from the exploration stage are used to construct a rough SUR based model of utility (VF). After this, whenever a state falls within a SUR certainty area, the sequence of SURs leading to the goal are followed, generating a new trace. The traces generated this way do not contain multivalued states. This implies that now the Trace Buffer contains well behaved samples that allow for feasible ANN based VF learning.

Algorithm 1 Search for tendencies

dec: decreasing tendency
i: current sensor
inc: increasing tendency
j: current trace episode
S: set of robot sensor values
T: trace

```
1: function CORRELATION_EVALUATOR (T)
2:for i ϵ S do
3:      inc is initially set to 1
4:      dec is initially set to 1
5:      for j ϵ T do
6:          if inc or dec then
7:              inc,dec←CheckCorrelationType(T, i, j)
8:      if inc then
9:          AddIncreasingTendencyTrace (T, i, j)
10:     else
11:         if dec then
12:             AddDecreasingTendencyTrace(T, i, j)
```

Consequently, the third stage of the approach proposed here implies learning a final multidimensional ANN based Value Function. This is mandatory if we aim to reach the goal in an optimal way. SURs can lead to the goal in a Manhattan like manner, one dimension at a time. However, for the sake of efficiency, in cases where there exist multidimensional dependencies between variables, or if we require a precise utility prediction, a multidimensional model is required. To this end, using the traces in the Trace Buffer as the training set, we have trained on-line an ANN based VF in the form of a multilayer perceptron by means of the Adam optimizer [21], which is an on-line learning method based on stochastic gradient descent. The batch size for the Adam algorithm can be adjusted, but the best results were obtained with a variable batch size equal to the trace length. Therefore, every time a new trace is obtained, a batch is created and the ANN is trained for a predefined number of epochs, as will be shown in section 4.

As commented at the beginning of the section, the three stages (exploration, learning SURs and learning ANN based VFs) are run concurrently. Exploration is active whenever a state is not within the certainty region of a SUR or VF. SUR and ANN learning take place whenever new traces leading to the goal are registered in the Trace Buffer. However, in this developmental approach the important point is that, *in the initial steps of learning, only the SURs are used to evaluate the*

candidate states, so the traces that are obtained do not contain ambiguities. These traces are used for training online the ANN that represents the multidimensional VF. This VF will control state evaluation when two criteria are met:

1) The goal is reached a predefined number of times G_a. This criterion is established once the SURs are assumed to be consistent, and consequently, the Trace Buffer contains relevant traces to reach the goal.
2) The prediction error of the ANN based VF is lower than a threshold e_u. This criterion is established to verify that the multidimensional space is learnable.

Once these two criteria are satisfied, the multidimensional VF becomes active and from this moment onwards, the traces are generated by it. That is, the Trace Buffer will start to contain samples with possible variations in all the dimensions, and the ANN based VF learning could degrade. To deal with this issue, e_u is continuously checked, and if it increases above a threshold, the SURs take control again.

Finally, Fig. 2 shows a diagram summarizing the different stages of the proposed methodology, as well as the criteria for alternating between them.

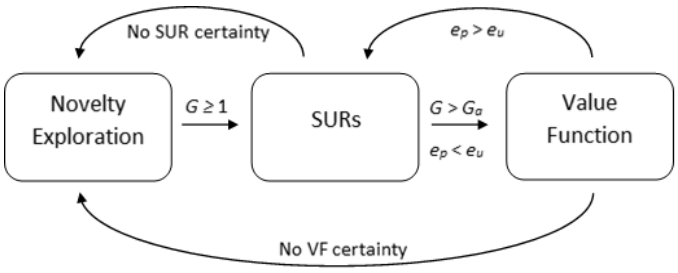

Fig. 2. Flow diagram representing the different stages of the methodology. G is the number of times the goal has been reached, whereas e_p is the prediction error of the ANN based VF.

IV. EXPERIMENTAL RESULTS

To illustrate the validity of the approach commented above in the developmental learning of VFs, a real robotic experiment has been designed, called "cleaning the playground". The experimental setup is displayed in Fig. 3, and it includes a Baxter robot, a white table with a delimited red zone that simulates a playground, two different types of objects (fruits and bricks) and two boxes, one brown and the other one transparent. The final objective is to collect the different objects and, according to their type, place them in the corresponding box without any prior knowledge. Consequently, in this setup, the two boxes make up two goals that must be discovered.

This setup works over a state space generated by three sensors: two provide the distance from the top corners of the playground to the robot (*dc1* and *dc2*), and the other one provides the type of object the robot is grabbing (t). Thus, at each iteration, the system will perceive a sensorial state: *S(t) = (dc1, dc2, t)*. As for the actions *A(t)*, they control the movement of the right arm of the Baxter robot, that is, the change in the direction of movement of the Baxter effector at a constant height and at a fixed speed. The position of the corners is detected in a calibration stage before placing the boxes on the table, so it does not have to be sensed unless the red zone is modified. This is why the boxes can be placed on the top of these corners, as displayed in Fig. 3.

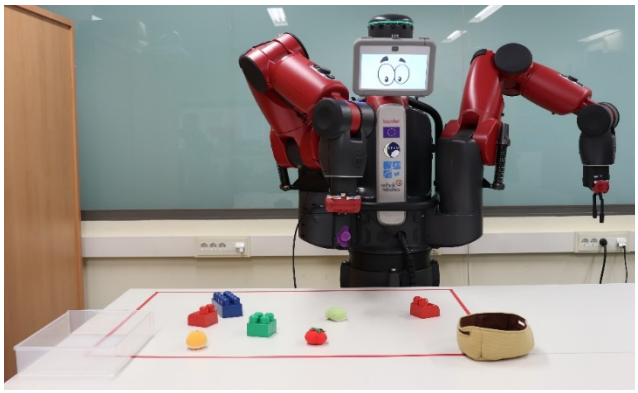

Fig. 3. The cleaning playground setup.

The robot operation starts with an object in the gripper (as shown in Fig. 3). When it is placed over any of the boxes, the object is automatically dropped, and the robot receives a reward (utility). This event triggers a reset of the scenario and the robot arm will be placed in a location over the table again, carrying a different type of object. As commented above, the state space in this experiment presents two goals. One of them is achieved when releasing the fruits in the brown box and the other is related to releasing the bricks in the transparent box. No utility is provided for any other point of the space.

A. Easy Cleaning

In a first setup (Fig. 3), the robot must learn to place the objects in their boxes when the latter are located in the upper corners of the playground area. Therefore, in this case, there is a direct relationship between each of the goals and two of the sensors of the robot (since the distance to the corners of the table are known).

TABLE I. PARAMETERIZATION OF THE DEVELOPMENTAL LEARNING IN THE EXPERIMENTS

ANN VF activation	**Value**
G_a	500
e_u	0.5
ANN Parameter	**Value**
Input neurons	3
Output neurons	1
Hidden layers	[10, 3]
Batch size	Trace length
Training epochs	10

As explained in section 2, initially, the robot has no idea where the goals are or how to reach them. Consequently, the exploratory behavior of a C-Drive is activated, and novelty guides the robot response, trying to find any goal. Once the first goal is reached and thus the first trace is obtained, VF learning starts, which implies the concurrent execution of the SURs creation algorithm and the online ANN learning process using the traces obtained by the SURs. The parameters used in this

learning process are displayed in Table 1, where the 3 input neurons of the ANN are the values of the 3 sensors that make up the state space, while the output is the expected utility.

Fig. 4 shows the results obtained for 10 independent runs of this experiment using the proposed methodology. Each line displays the accumulated reward (goal achievements) obtained through the iterations for each of the executions of the experiment. The moment when the VF replaces the SURs is marked by an increase in the thickness of the line. In this case, since the problem is relatively simple, there is almost no improvement in the efficiency of the robot when solving the task once the VF takes control. However, what is relevant is that the VF learning did not degrade starting from traces created with the SURs and then continuing with those produced by the VF.

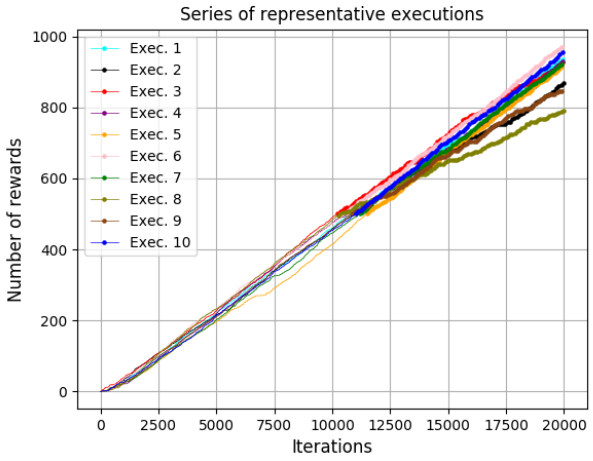

Fig. 4. Performance analysis for the first setup.

To illustrate the problem of learning the VF from scratch, Fig. 5 shows 10 executions of this simple setup in which the SURs creation was disabled. That is, the system tries to learn the VF directly from the traces it generates. As it can be observed in the figure, the VF is never properly learnt due to the expected ambiguities in the traces (it only reaches 250 rewards in 20000 iterations as compared to more than 800 in the previous case).

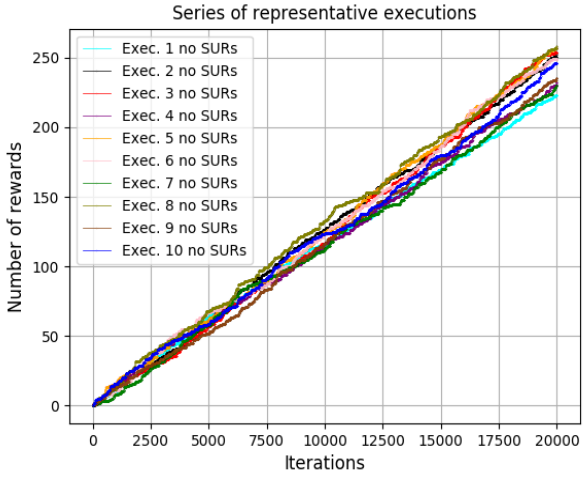

Fig. 5. Performance analysis for the first setup without using SURs.

B. *Complex cleaning*

In this second setup (Fig. 8), the boxes are placed at intermediate points in the playground area. This causes the complexity of the problem to increase, since the robot does not have a direct perception of their position. Therefore, it will have to learn to triangulate the position from the corners of the playground. The parameters used in the VF learning process are the same as in the previous case (Table 1).

As in the previous case, Fig. 6 shows the time required for the correct learning of the VF in 10 executions of the experiment. In this case, it can be clearly seen how the slope of the curves increases, reflecting how the number of rewards per unit of time grows with iterations. This represents an increase in the system's efficiency as it improves its knowledge of the scenario and the task to be solved. It can also be seen how the efficiency of the system is higher when the VF takes the control (the slope of the graph increases) in most of the cases, which means that the developmental strategy is successful in this type of complex setup, as expected.

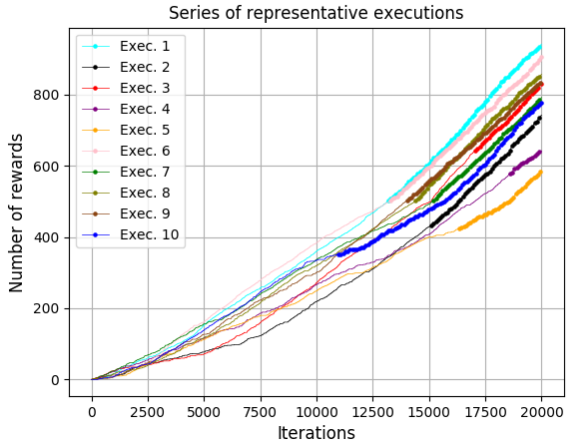

Fig. 6. Performance analysis for the second setup.

Once at this point, to see in a clearer way that the VF improves the efficiency of the robot when solving the task, an analysis of the traces of the robot to reach the final objectives using both approaches (SURs and VF), is carried out. To this end, Fig. 7 shows a representation of 100 random starting points (in which the robot starts by holding the object) ordered by increasing distance from that point to the goal, and the number of iterations necessary to reach the goal using SURs and VF. It is easy to appreciate in the figure how the number of steps (iterations) necessary to solve the problem with SURs is larger than those required by the VF. This change in efficiency is mainly due to the fact that the VF is able to generalize better over the state space in multidimensional problems.

Finally, Fig. 8 contains a representation of the trajectory followed by the real robot arm to reach the goal using SURs (top image) and VF (bottom image), in a representative execution of the experiment. As it can be observed by following the yellow arrows, when using the SURs, the trajectory is not optimal and the robot first moves the object towards the corner, and then towards the corresponding box. This is the expected result, because the robot is following only one-dimensional paths. In

the bottom image, it can be clearly observed that now the movement is optimal, as expected when a multidimensional model in the form of a VF is considered.

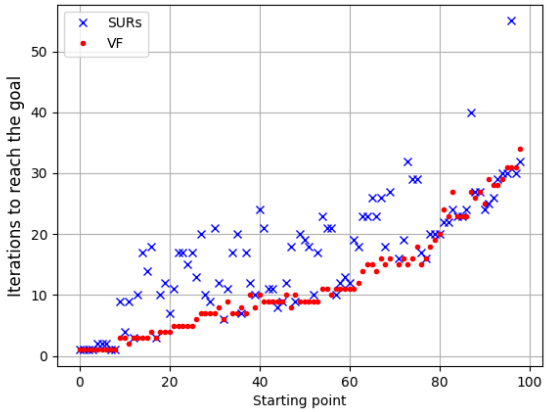

Fig. 7. Comparison of the time needed to solve the task using SURs and using Value Function .

With these simple results, the potential of the proposed approach for the developmental learning of Value Functions becomes evident. As shown, it allows us to take advantage of the best of each of the utility modeling techniques (SURs and VFs). Thus, depending on the complexity of the problem, the use of the different elements will vary to allow the utility to be modeled as efficiently as possible. Finally, the need to learn the VF and not only solve the problem using SURs has been justified, paying attention to the efficiency obtained in the behavior of the robot.

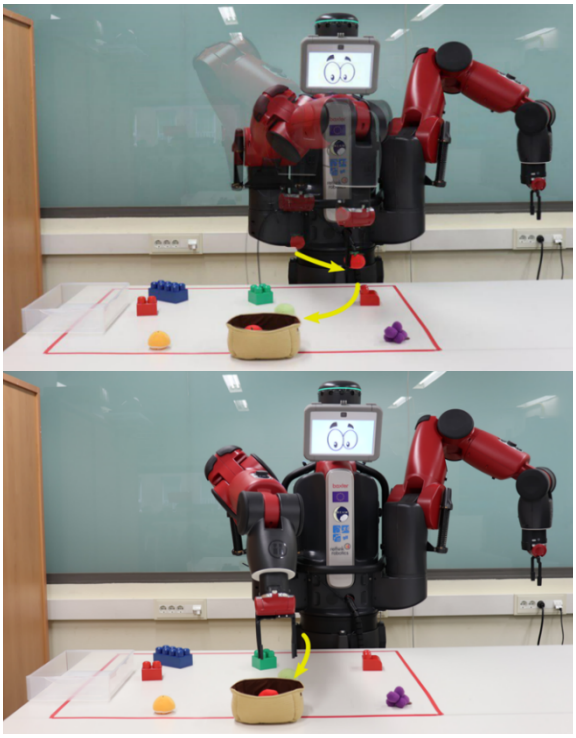

Fig. 8. Comparison between real robot traces before and after learning the VF. Top: Trajectory using SURs. Bottom: Trajectory using VF.

V. CONCLUSIONS

This paper has addressed the issue of learning Value Functions (VF) on-line in the context of a motivational mechanism for open-ended learning robots. The main issue that has been considered is the problem of ambiguities, due to the necessary use of exploratory motivations, in the values of the points in the traces obtained towards the goal and which are used in order to train ANNs to represent the VFs.

We have analyzed the possibility of the concurrent application of a simpler, albeit less efficient, SUR based approximation to VF production. The rationale behind this is that SURs are much faster to obtain once a goal is found, thus reducing the reliance on exploratory motivations to construct the VF. Using the SURs to reach the goal produces much better configured trace sets with less ambiguities in order to obtain the ANN based VFs. The experimental results show that this is indeed the case. In examples where just using traditional VF learning with exploratory motivations provide very poor results, using the concurrent application of SURs and ANN based VF training permits reaching very satisfactory models.

We are currently implementing these strategies in experiments that make use of the whole e-MDB cognitive architecture, including its memory elements to evaluate the efficiency of the reuse of these VFs when facing new domains.

ACKNOWLEDGMENT

This work has been partially funded by the EU's H2020 research programme (grant No 640891 DREAM), Ministerio de Ciencia, Innovación y Universidades of Spain/FEDER (grant RTI2018-101114-B-I00), Xunta de Galicia and FEDER (grant ED431C 2017/12), and by the Spanish Ministry of Education, Culture and Sports for the FPU grant of Alejandro Romero.

REFERENCES

[1] A. G. Barto, “Intrinsic motivation and reinforcement learning,” in *Intrinsically Motivated Learning in Natural and Artificial Systems*, 2013.

[2] S. Doncieux *et al.*, “Open-ended learning: a conceptual framework based on representational redescription,” *Front. Neurorobot.*, 2018.

[3] G. Baldassarre and M. Mirolli, “Intrinsically motivated learning in natural and artificial systems,” in *Intrinsically Motivated Learning Systems: an Overview*, Springer Berlin Heidelberg, 2013, pp. 1–14.

[4] A. Romero, A. Prieto, F. Bellas, R. J. Duro, “Simplifying the creation and management of utility models in continuous domains for cognitive robotics”, *Neurocomputing*, Vol 353, pp. 106-118, 2019

[5] R. S. Sutton and A. G. Barto, “Introduction to Reinforcement Learning,” *Learning*, vol. 4, no. 1996, pp. 1–5, 1998.

[6] X. Huang and J. Weng, “Value system development for a robot,” *IEEE Int. Conf. Neural Networks - Conf. Proc.*, vol. 4, pp. 2883–2888, 2004.

[7] K. E. Merrick, “A Comparative Study of Value Systems for Self-Motivated Exploration and Learning by Robots,” *IEEE Trans. Auton. Ment. Dev.*, vol. 2, no. 2, pp. 1–15, 2010.

[8] J. A. Starzyk, “Motivated Learning for Computational Intelligence,” *Comput. Model. Simul. Intellect ...*, no. Ml, pp. 265–292, 2010.

[9] N. Hawes, “A survey of motivation frameworks for intelligent systems,” *Artif. Intell.*, vol. 175, no. 5–6, pp. 1020–1036, 2011.

[10] X. Huang and J. Weng, “Novelty and Reinforcement Learning in the Value System of Developmental Robots,” *Proc. Second Int. Work. Epigenetic Robot.*, pp. 47–55, 2002.

[11] A. Baranes and P. Y. Oudeyer, “Active learning of inverse models with intrinsically motivated goal exploration in robots,” *Rob. Auton. Syst.*, vol. 61, no. 1, pp. 49–73, 2013.

[12] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrel, and A. Efros, "Large-scale study of curiosity-driven learning," *arXiv Prepr.*, no. 1808.04355, 2018.

[13] K. Seepanomwan, V. G. Santucci, and G. Baldassarre, "Intrinsically Motivated Discovered Outcomes Boost User's Goals Achievement in a Humanoid Robot," in *Proc. 2017 ICDL-EpiRob*, 2017, pp. 178–183.

[14] P. Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? A typology of computational approaches," *Front. Neurorobot.*, vol. 1, p. 6, 2009.

[15] U. Nehmzow, Y. Gatsoulis, E. Kerr, J. Condell, N. Siddique, and T. M. Mcginnity, *Intrinsically Motivated Learning in Natural and Artificial Systems*. 2013.

[16] R. M. Ryan and E. L. Deci, "Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions," vol. 67, pp. 54–67, 2000.

[17] G. Baldassarre and M. Mirolli, "Deciding which skill to learn when: Temporal-difference competence-based intrinsic motivation (TD-CB-IM)," *Intrinsically Motiv. Learn. Nat. Artif. Syst.*, pp. 257–278, 2013.

[18] A. Romero, F. Bellas, J. A. Becerra, and R. J. Duro, "Bootstrapping Autonomous Skill Learning in the MDB Cognitive Architecture," *Lect. Notes Comput. Sci.*, vol. 11486, pp. 120–129, 2019.

[19] A. Prieto, A. Romero, F. Bellas, R. Salgado, and R. J. Duro, "Introducing Separable Utility Regions in a Motivational Engine for Cognitive Developmental Robotics," Integr. Comput. Aided. Eng., vol. 26, no. 1, pp.3-20, 2019

[20] R. Salgado, A. Prieto, F. Bellas, L. Calvo-Varela, and R. J. Duro, "Motivational engine with autonomous sub-goal identification for the Multilevel Darwinist Brain," *Biol. Inspired Cogn. Archit.*, vol. 17, 2016.

[21] Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization, cite arxiv:1412.6980.