An Enactive Approach to Autonomous Agent and Robot Learning

Olivier L. Georgeon, Christian Wolf, Simon Gay

An Enactive Approach to Autonomous Agent and Robot Learning

Olivier L. Georgeon
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205
Villeurbanne, F-69622 France
olivier.georgeon@liris.cnrs.fr

Christian Wolf
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
Villeurbanne, F-69621 France
christian.wolf@liris.cnrs.fr

Simon Gay
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205
Villeurbanne, F-69622 France
simon.gay@liris.cnrs.fr

***Abstract*— A novel way to model an agent interacting with an environment is introduced, called an Enactive Markov Decision Process (EMDP). An EMDP keeps perception and action embedded within sensorimotor schemes rather than dissociated. Instead of seeking a goal associated with a reward, as in reinforcement learning, an EMDP agent is driven by two forms of self-motivation: successfully enacting sequences of interactions (autotelic motivation), and preferably enacting interactions that have predefined positive values (interactional motivation). An EMDP learning algorithm is presented. Results show that the agent develops a rudimentary form of self-programming, along with active perception as it learns to master the sensorimotor contingencies afforded by its coupling with the environment.**

Keywords—Enaction; self-motivation; constructivist learning.

I. INTRODUCTION

In cognitive science, there has been a customary and traditional tripartite division of the mind between perception, the control system, and motor action. This view has been nicely dubbed the "classic sandwich model" by Susan Hurley [1]. There have, nonetheless, been many attempts to challenge this traditional picture, particularly in the field of robotics [2][3], but also from a more psychological and theoretical perspective [4][5]. In particular, the idea emerged that it might be a mistake to consider sensation independently from action and that we should design cognitive systems on the basis of *low-level sensorimotor loops* that represent sensorimotor patterns of interaction. Here, we introduce a modeling approach that goes a step beyond the notion of low-level sensorimotor loops by simply considering sensorimotor patterns—also called *sensorimotor schemes* by Piaget [6]—as the atomic elements manipulated by our algorithms.

Furthermore, the theory of enaction [7] stresses the importance of *constitutive autonomy*—the capacity of the system to "self-constitute its identity" [8]. Our modeling approach offers a way to address the problem of constitutive autonomy by supporting a form of autonomous sensorimotor self-programming through interaction with the environment.

This work was supported by the French Agence Nationale de la Recherche (ANR) contract ANR-10-PDOC-007-01. We gratefully thank James Marshall and Riccardo Manzotti for their contribution to this report.

II. ENACTIVE MARKOV DECISION PROCESS (EMDP)

We introduce a new way to model an agent interacting with an environment called an Enactive Markov Decision Process (EMDP). An EMDP conflates action and observation within a single entity called an *interaction*. Based on the *EMDP model*, we define *EMDP problems* consisting of designing agents that learn to master their sensorimotor contingencies [9] so as to exhibit self-motivation. The EMDP model allows defining two forms of self-motivation: the motivation to be *in control* of one's own activity by seeking to successfully enact interactions, and the motivation to enact interactions that have predefined positive values and to avoid enacting interactions that have predefined negative values. We call the former *autotelic motivation* in reference to Steel's autotelic principle [10], and the latter *interactional motivation* [11].

A. EMDP Formalism compared to POMDP

Formally, we define an EMDP as a tuple (S, I, q, v) in which S is the set of environment states; I is the set of *primitive interactions* offered by the coupling between the agent and the environment; q is a probability distribution such that $q(s_{t+1}|s_t,i_t)$ gives the probability that the environment transitions to state $s_{t+1} \in S$ when the agent chooses interaction $i_t \in I$ in state s_t; and v is a probability distribution such that $v(e_t|s_t,i_t)$ gives the probability that the agent receives the input $e_t \in I$ after choosing i_t in state s_t. We call i_t the *intended interaction* because it represents the sensorimotor scheme that the agent intends to enact at the beginning of step t; and e_t the *enacted interaction* because it represents the sensorimotor scheme that the agent records as actually enacted at the end of step t. If the enacted interaction equals the intended interaction ($e_t=i_t$) then the attempted enaction of i_t is considered a *success*, otherwise, it is considered a *failure*. The EMDP cycle evolves as follows and is illustrated in Fig. 1.

1. Let $t = 0$ and let $s_0 \in S$ denote the initial environment state.
2. Choose intended interaction $i_t \in I$.
3. Transit to new state $s_{t+1} \in S$ with probability $q(s_{t+1}|s_t,i_t)$.
4. Generate *actually enacted* $e_t \in I$ with probability $v(e_t|s_t,i_t)$.
5. $t = t + 1$.
6. Goto 2.

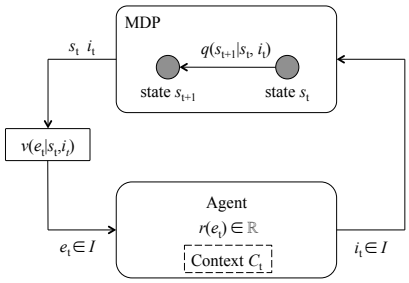

Fig.1. Diagram of an Enactive Markov Decision Process (EMDP). At time t, the agent chooses an *intended primitive interaction* i_t (the agent's output). The attempt to enact i_t generates a transition of the environment (represented by the Markov Decision Process, MDP) from state s_t to s_{t+1}. The agent then receives the *enacted primitive interaction* $e_t \in I$ (the agent's input). If $e_t = i_t$, then the attempt to enact i_t is considered a *success*, otherwise, a *failure*. The agent's "perception of its environment" is an internal construct C_t rather than the input e_t.

There are three major differences between an EMDP and a Partially Observable Markov Decision Process (POMDP) [12]: (a) the cycle does not start from the environment but from the agent, implying a conceptual inversion of the perception/action cycle (as some other authors have also proposed, e.g. [13]); (b) The agent's input and output belong to the same set I rather than two different sets (observations O and actions A); (c) There is no reward defined as a function of the MDP's state.

To pursue the comparison, it would be possible to decompose the interaction set as the product space of a POMDP's action set and observation set ($I = A \times O$). Yet, the philosophy of an EMDP differs in that the observation set would contain too little information about the state s to allow traditional reinforcement learning techniques to generate interesting behaviors. To illustrate this difference, we present an experiment in which the equivalent observation set is reduced to a single bit, while the interaction set if fairly sophisticated. The lack of observational information is counterbalanced by the fact that the observation's meaning depends on the previous action. Moreover, the EMDP goal is not of maximizing a reward function but rather of exhibiting a more complex form of self-motivation that we present next.

B. Self motivation

Now that we have formally defined a *successful enaction*, we can define *autotelic motivation* as the tendency to learn to successfully enact interactions. This tendency involves neither a reward nor a goal; it is intrinsically encoded in the algorithm through discovering, recording, and re-enacting sequences of interactions that capture regularities in the coupling with the environment, as we will explain in Section III.

Additionally, we define interactional motivation as a way to associate an "innate" intrinsic value with some interactions, for example: a positive value with interactions representing ingesting food, and a negative value with interactions representing getting hurt. Interactional motivation is meant to underdetermine the agent's behavior so that the agent can use neutral interactions to place itself in situations in which it can successfully enact positive interactions and to stay away from situations in which negative interactions cannot be avoided. We specify interactional motivation through a function r: $I \rightarrow \mathbb{R}$ that associates a scalar value $r(e)$ with each primitive interaction $e \in I$. Interactional motivation relates to *intrinsic motivation* (e.g., [14]) in that it is defined independently of the environment's state.

To fulfill both its autotelic and interactional motivation, we expect an EMDP agent to discover, memorize, and exploit regularities that exist in its coupling with its environment. Section II.C formally defines a regularity of interaction as a series of primitive interactions that can be learned and enacted as a whole sequence.

C. Self-programming EMDP agents

We define a composite interaction as a sequence of two interactions $i_c = \langle i_{pre}, i_{post} \rangle$, where i_{pre} and i_{post} may be primitive interactions or other composite interactions. We refer to i_{pre} as i_c's *pre-interaction*, also denoted $pre(i_c)$, and to i_{post} as i_c's *post-interaction*, also denoted $post(i_c)$. We define K_t as the set of composite interactions known by the agent at time t, and $J_t = I \cup K_t$ as the set of all interactions, primitive or composite, at time t. Interactions in J_t are thus hierarchically organized in a pairwise manner, all the way down to primitive interactions. We extend r to be a function from $J_t \rightarrow \mathbb{R}$ that gives the motivational value of a composite interaction as the sum of the values of its primitive interactions, meaning that enacting a composite interaction has the same motivational value as enacting all of its primitive interactions.

A self-programming agent can choose to enact any interaction in J_t, primitive or composite. We call the mechanism that chooses an interaction the *decisional mechanism*, the point in time t_d when this choice is made a *decision time*, and the time lapse during which an interaction is enacted a *decision cycle*. Decision steps thus do not occur on each time step but rather between each decision cycle. Trying to enact a composite interaction i_c consists of sequentially trying to enact the series of the k primitive interactions $\langle i_{p1} \ldots i_{pk} \rangle$ that compose i_c over the next k time steps $t_d+1, \ldots t_d+k$. If all the primitive interactions are successfully enacted, then i_c is successfully enacted, and the decision cycle ends at time t_d+k. If the enaction of the j^{th} element of i_c fails, then the decision cycle is interrupted at time t_d+j and the actually enacted composite interaction is constructed from the series of the j actually enacted primitive interactions $\langle e_{p1} \ldots e_{pj} \rangle$.

We describe such agents as self-programming because composite interactions work as programs that the agent learns and subsequently executes. Rather than being written in a conventional programming language, such programs are written in the "agent's programming language" in the sense that they are made of sequences of instructions that the agent knows how to execute.

III. Interactional Motivation System (IMOS)

We implemented a new algorithm to control EMDP agents, called Interactional MOtivation System (IMOS), based on a previous algorithm [16]. Section III.A provides a new explanation of the main mechanisms of the previous algorithm

in the light of the EMDP formalism introduced in this paper: the learning mechanism and the context construction mechanism. We refer the reader to the previous paper [16] for a more thorough explanation. Section III.B presents the new decisional mechanism implemented in IMOS.

A. Learning and context construction mechanism

The agent's current situation is represented by a set of interactions $C_d \subset J_d$ referred to as the *context* (from now on, all "X_t" at decision time t_d is denoted "X_d"). $C_0 = \emptyset$ and $J_0 = I$. At the end of decision cycle t_d, the agent records or reinforces the composite interactions whose pre-interaction belongs to C_d and whose post-interaction is the actually enacted interaction e_d. The set of learned or reinforced composite interactions is thus $L_d = \{\langle i_{pre}, e_d\rangle \mid i_{pre} \in C_d\}$. Each learned or reinforced composite interaction thus records the experience that, in a context in which i_{pre} was enacted, e_d was subsequently enacted. The set of composite interactions known by the agent at decision cycle t_{d+1} thus becomes $K_{d+1} = K_d \cup L_d$. Let $wei_t(i) \in \mathbb{N}$ be the weight of composite interaction i at time t. The weight of a new learned composite interaction is initialized to 1. Reinforcing an already-known composite interaction means incrementing its weight.

If i_d's enaction failed ($e_d \neq i_d$), then the agent adds e_d to the set of i_d's *alternative interactions* $alt_d(i_d) \subset J_d$ known at time t_d. Later, the decisional mechanism uses $alt_d(i_d)$ to evaluate the odds that other interactions may be actually enacted when trying to enact i_d again, as we further explain in Section III.B.

Before making the next decision at time t_{d+1}, the agent computes the context C_{d+1} as the set of interactions that were enacted at the end of decision cycle t_d and that are sufficiently reinforced. We define a threshold η, and let $L\eta_d = \{l \in L_d \mid wei_d(l) > \eta\}$ be the set of composite interactions in L_d whose weight is greater than the threshold η at time t_d.

$$C_{d+1} = L\eta_d \cup \{e_c\} \cup \{post(e_c)\} \quad (1)$$

Equation (1) implies that C_{d+1} contains interactions of various lengths that have just been enacted at the end of decision cycle t_d. Since, over time, the agent tends to enact interactions that capture sequential regularities afforded by the environment, C_d tends to represent the agent's situation in terms of the sequences of interactions that are the most representative of the situation at time t_d.

In the experiments reported in Sections IV and V, we set $\eta = 5$, meaning that a sequence had to be encountered 5 times before being eligible for inclusion in the context. Lower values of η tend to favor faster learning of possibly less positive behaviors; greater values of η tend to favor slower learning of possibly more positive behaviors. Automatically adjusting η is still an open question. Limiting the size of the context to interactions sufficiently reinforced prevents combinatorial growth of the number of interactions learned over time. With this mechanism, experiments show that the number of interactions learned after each decision cycle remains limited (less than 10 in the experiment reported in Section 5).

B. New decisional mechanism

At decision cycle t_d, let $A_d = \{a \in K_d \mid pre(a) \in C_d\}$ be the set of composite interactions whose pre-interaction belongs to the current context, called *activated interactions*. Let $P_d = \{p \in J_d \mid \exists\ a \in A_d,\ p = post(a)\}$ be the set of interactions that constitute a post-interaction of an activated interaction. We call this set the *proposed interactions*. The agent thus anticipates that proposed interactions might be successfully enacted in the current context based on the regularities the agent has learned thus far. On decision cycle t_d, the agent chooses the next intended interaction i_d from amongst the proposed interactions in P_d, as we explain next.

For each proposed interaction p in P_d, let $A_{pd} \subset A_d$ be the set of activated interactions that proposed p. The decisional mechanism computes a *proclivity value* $proc_d(p)$ as the sum of the weight of all activated interactions a in A_{pd} multiplied by the value $r(p)$. The agent thus has a proclivity to try to enact the interactions that are proposed by the most reinforced activated interactions and that have the highest motivational values. Interactions that have a negative value create a negative proclivity, which restrains the agent from trying to enact them.

$$proc_d(p) = \sum_{a \in A_{pd}} wei_d(a) \times r(p) \quad (2)$$

Additionally, the decisional mechanism computes an alternative proclivity value $proc_d(e)$ for each alternative interaction $e \in alt_d(p)$ that may result from the attempt to enact p. For each alternative interaction e, we let A_{ed} denote the set of activated interactions that proposed e.

$$proc_d(e) = \sum_{a \in A_{ed}} wei_d(a) \times r(e) \quad (3)$$

The final proclivity $proc\_final_d(p)$ is then set equal to the sum of the proclivity of p and of the proclivities of all of its alternative interactions. Alternative interactions of p that have a negative motivational value thus restrain the agent from trying to enact p.

$$proc\_final_d(p) = proc_d(p) + \sum_{e \in altd(p)} proc_d(e) \quad (4)$$

The decisional mechanism then chooses the intended interaction i_d as the interaction in P_d that has the highest final proclivity. As a result, the agent intends to enact the interaction i_d that has the highest positive motivational value $r(i_d)$, weighted by the agent's confidence that i_d will be successfully enacted in the current context C_d, and balanced by the weighted "fear" that it may result into a negative value $r(e_d) < 0$ if i_d's enaction fails. Because composite interactions may contain interactions with negative values, this mechanism does not drive the agent to the highest immediate value but rather allows the agent to enact sequences of negative interactions to reach even greater positive interactions, and to avoid immediate positive interactions that likely would lead subsequently to even more negative interactions.

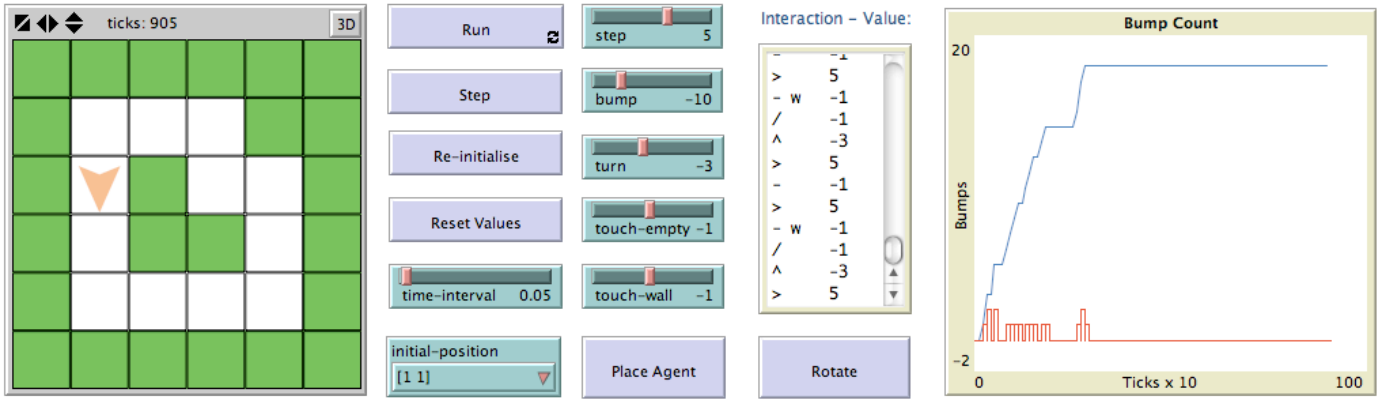

Fig. 2. The Small Loop Problem (SLP) in NetLogo. The environment (left) is the loop of white squares surrounded by green walls. The agent (brown arrowhead) can try to move one cell forward, turn to the left or to the right, feel in front, to the left or to the right, but it ignores the meaning of interactions. The experimenter can preset the values of the primitive interactions using the slider controls (center). The *Interaction-Value* window shows ASCII codes representing the primitive interactions enacted by the agent over time next to their values. The *Bump Count* graph (right) displays the number of times the agent bumps into a wall (cumulative total in blue), showing that the agent gradually learns to avoid bumping into walls. When the agent touches/feels a cell, the cell flashes yellow, and when the agent bumps into a wall, the wall flashes red, making the agent's behavior intelligible to the experimenter.

The main novelty of IMOS resides in the fact that it takes into account the various alternative interactions that may result from intended interactions, whereas the previous algorithm only considered their success or failure. The algorithm is also clearer in terms of the new EMDP formalism.

IV. Experiment in a Simulated Environment

Cognitive theories have often advocated that hierarchical sequence learning is a key feature of cognition (e.g., [15]). Accordingly, we designed an EMDP problem that offers hierarchical sequential regularities of interaction, called the Small Loop Problem (SLP) [17] (Fig. 2). The possibilities of interaction are summarized in Fig. 3a. Examples hierarchical regularities of interactions are provided in Fig. 3b (induced by the loop-shaped pathway that constrains the agent's behavior).

The *feel* interactions especially illustrate the enactivist approach: the agent only receives information about the environment through the enaction of interactions. For example, trying to enact i_7 informs the agent about the absence (if it succeeds) or the presence (if it fails and results in i_8) of a wall in front, but the agent initially ignores this meaning, as well as the position of interactions and the existence of walls. In contrast, the turn interactions (i_3 and i_4) always succeed when the agent tries to enact them, and, therefore, provide no information (hence the need for representing the situation not only upon the last enacted interaction but upon longer sequences).

The SLP is deterministic, making it a particular EMDP in which q and v implement no stochasticity. The agent is nonetheless confronted with uncertainty because it cannot initially predict the consequences of its intended interactions until it starts learning the regularities afforded by the environment. For example, when circling the loop counterclockwise, if it feels a wall in front, it can often feel an empty cell to the left, but not always. The agent's algorithm is also deterministic, meaning that two runs lead to the same behavior. Different behaviors can nonetheless be observed by starting the agent from different initial positions.

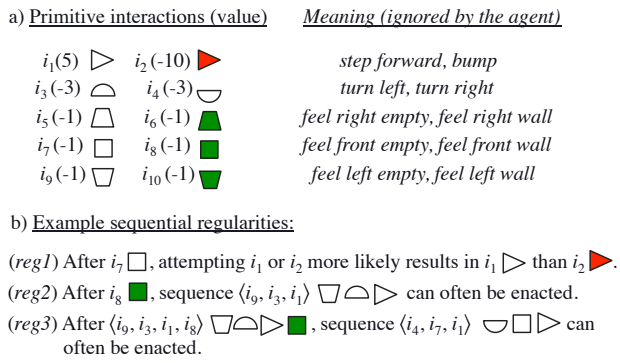

Fig. 3. Interactions offered by the SLP. a) The agent has 10 primitive interactions at its disposal, which have scalar values set by the experimenter (in parentheses). Interactions that are afforded by empty cells are represented in white and those afforded by walls in green and red, but the agent ignores this distinction. To the agent, interactions only differ by their values. b) The coupling offers hierarchical sequential regularities of interactions. For example, we expect the agent, in discovering and exploiting (*reg1*), to choose interaction i_7, and if this effectively results in i_7, to subsequently choose i_1 so as to safely enact i_1 which has a positive value, thus avoiding i_2 which has a very negative value. Regularities have a hierarchical structure: for example, $\langle i_9, i_3, i_1 \rangle$ in (*reg2*) is a subsequence of the (*reg3*) sequence $\langle i_9, i_3, i_1, i_8 \rangle$. The agent can begin by discovering and exploiting lower-level regularities, then learn higher-level regularities from sequences of lower-level regularities.

The experiment can be run interactively online[1]. A video showing an example run is also available[2], corresponding to the trace reported in Fig. 4. Results show that the agent generally learns to avoid bumping into walls by adopting the behavior of feeling in front before trying to move forward within the first 300 steps (Bump Count graph in Fig. 3). Then, when the agent feels a wall in front, it progressively learns to feel to the side before deciding on which direction to turn. This behavior generally leads the agent to start to indefinitely circle the loop after approximately 400 steps, as reported in the example in Fig. 4. In some instances, it learns suboptimal

[1] http://e-ernest.blogspot.fr/2012/03/small-loop-challenge.html
[2] http://www.youtube.com/watch?v=yliUE39l2R4

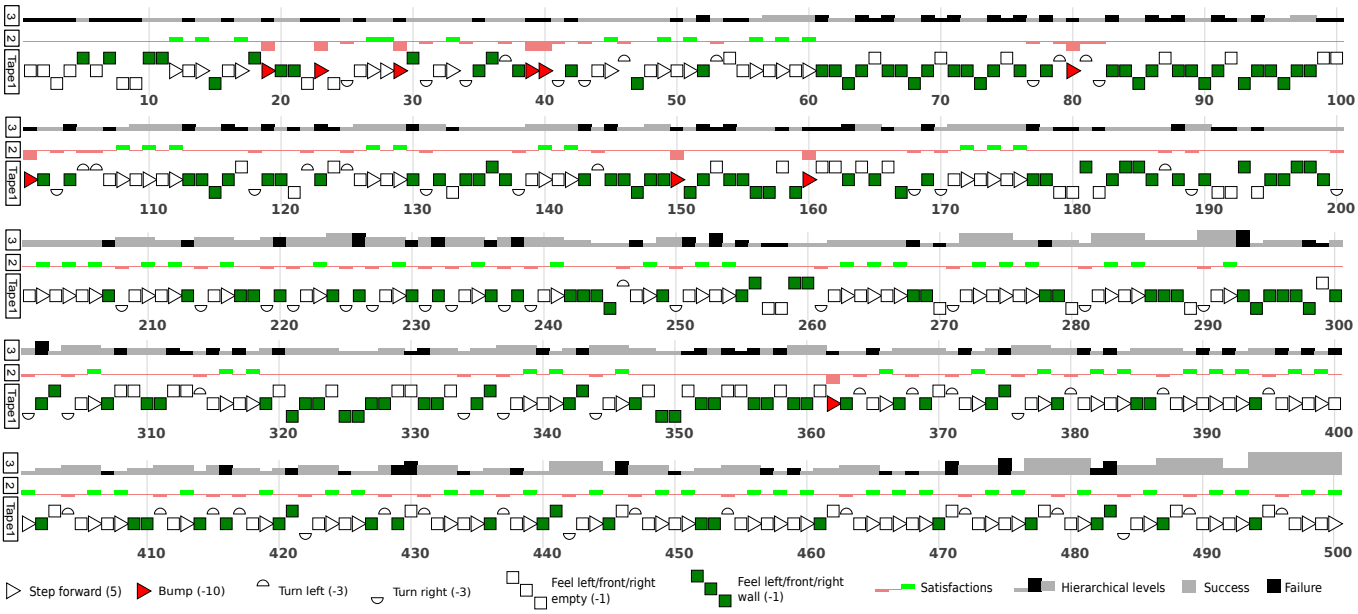

Fig. 4. First 500 steps of an example run of an IMOS agent in the SLP. Tape 1 represents the primitive interactions actually enacted over time: *step forward* (white triangles), *bump* (red triangles), *turn* (half-circles), *feel empty* (white squares), *feel wall* (green squares). Tape 2 represents the values of the enacted primitive interactions as a bar graph (green when positive, red when negative). Tape 3 represents the level of the enacted composite interaction in the hierarchy (gray when the enaction of the primitive interaction was successful, black when it failed, thus interrupting the decision cycle). The second-order composite interaction *feel front empty – move forward* (exploiting *reg1*) was successfully enacted for the first time on steps 57-58, then again on steps 59-60. The agent unsuccessfully tried to enact it on step 52 but the unexpected feeling of a wall caused its interruption, preventing the agent from bumping into the wall (black, second-order segment in Tape 3). Because the agent makes no assumptions on the meaning of primitive interactions, it also tries composite interactions that seem absurd to the observer (such as feeling twice to the front on steps 97-98) but subsequently abandons them because they prove useless to satisfy its interactional motivation. A third-order composite interaction was successfully enacted for the first time on steps 208-210: *turn right – feel front empty – step forward*, and its counterpart to the left on steps 404-406. The fourth-order composite interaction *feel left wall – turn right – feel front empty – step forward* was enacted on steps 441-444. From that point, the agent had learned enough regularities to circle the loop counterclockwise indefinitely by appropriately feeling in front and to the right to prevent bumping and turning to the wrong direction, thus also minimizing mildly negative *turn* interactions.

behaviors such as moving back and forth along a single edge of the loop, repeatedly making U-turns at each end of the edge. We observed that IMOS was generally faster at learning to avoid bumping into walls than the previous algorithm (e.g., 10 bumps in the example in Fig. 4 vs 18 bumps in the example reported in [17]).

V. Experiment with an e-puck robot

We ran a similar experiment with an e-puck robot [18] in the *Box world* shown in Fig. 5. The set of primitive interactions I was the same as those in the experiment in Section IV, with the same value function r. The interaction i_1 (*step forward*) consisted of moving approximately 5 cm. If the robot bumped into a wall during this displacement, the movement was interrupted and the interaction i_2 was actually enacted. The three infrared sensors available in the front, front-left, and front-right directions were used to detect bumping (using a range of approximately 0.5 cm). The interactions i_3 (*turn left*) and i_4 (*turn right*) consisted of spinning approximately 90° to the left or to the right. The *feel* interactions (to the front, left, or right) were implemented using the three infrared sensors available on the front, left, and right sides of the robot, using a range of approximately 5 cm. Trying to enact a feeling interaction i_5, i_7, or i_9 consisted of switching on the corresponding infrared LED and reading the infrared detection signal. Interactions i_5, i_7, i_9 ended up actually enacted if no wall was detected, and i_6, i_8, i_{10} were actually enacted if a wall was detected within the approximate 5 cm range respectively to the right, front, and left.

Fig. 5. The e-puck robot in the *Box world*.

Similar to the experiment in Section IV, this experiment showed that the robot learned to avoid bumping by using its front sensor to check for the absence of a wall before moving forward. A video of an example run is available online[3]. The first hundred steps are reported in the trace in Fig 6.

This experiment illustrates how an EMDP algorithm can be transferred to a robot interacting with the real world. The algorithm worked well with imprecise displacements and noisy sensors. In some cases, spurious composite interactions may be learned (e.g., *feel front empty* then *bump*) but they do not degrade the robot's behavior as long as correct composite interactions have dominant weights. The algorithm can be

[3] http://e-ernest.blogspot.fr/2012/04/ernest-7-in-e-puck.html

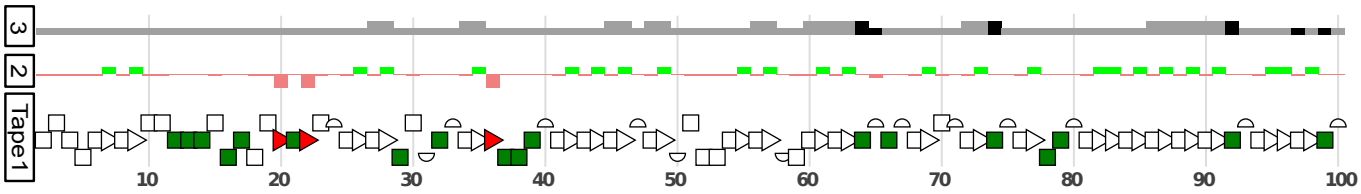

Fig. 6. First 100 steps of an example run of an e-puck robot controlled by IMOS in the *Box world*, using the same symbols as in Fig. 4. Tape 1 shows that the robot learned to avoid bumping (red triangles) after step 36. Tape 2 shows that the robot received more positive values more regularly from step 42 on. Tape 1 with Tape 3 show that the composite interaction *feel front empty – step forward* was enacted for the first time on steps 27-28. On step 64, 74, and 92, this composite interaction was interrupted (black second-order segment in Tape 3) due to the fact that the *feel front empty* intended primitive interaction resulted in the *feel front wall* interaction being actually enacted, preventing the robot from bumping and inciting it to turn left instead.

used as such to control the robot because the robot's motivation is defined in terms of interactions that are known to the robot itself, rather than requiring states of the world to be explicitly specified, as in POMDPs.

Note that this experiment rested on the fact that the ranges of the sensors were tuned to approximately correspond to the distance covered by a *step forward* move (5 cm). If these distances are too different, the robot learns that the sensors are useless. In this case, the robot moves forward without using the sensors and does not avoid bumping. Nonetheless, if we reduce the cost of turning to a small cost ($r(i_3) = r(i_4) = -1$), the robot learns to avoid bumping by alternating moving forward and turning, which causes it to circle around continuously in the box without hitting the walls.

VI. Conclusion

The EMDP model offers a framework for conceptualizing an agent's coupling with its environment without separating perception from action. *EMDP problems* consist of designing agents that tend to successfully enact sequences of interactions, and, simultaneously, seek to enact interactions that have predefined positive values while avoiding interaction that have strong negative values. While this objective involves a scalar value function, it differs from the reinforcement-learning objective by the fact that maximizing this value is not the only goal.

We introduce a new algorithm to address EMDP problems called IMOS. By learning sequences of behaviors, IMOS relates to *adaptive history* methods (e.g. [19]), used in reinforcement learning, but differs by the fact that the learning is not driven by a reward function of the MDP's state. By involving bottom-up chaining of behaviors, it relates to *means-end analysis* approaches (e.g., [20]), but differs by the fact that it is not based upon triples ⟨pre-observation, action, post-observation⟩, but rather couples ⟨pre-interaction, post-interaction⟩, allowing the recursive learning of hierarchies of sequences, and rudimentary self-programming.

We propose a method to assess EMDP agents through behavioral analysis. In our experiments, IMOS agents exhibit autotelic motivation and interactional motivation, in particular by learning to use some sensorimotor interactions (feeling) as a form of active perception to subsequently choose adapted behaviors (although the specific meaning of the interactions is initially unknown to the agent).

References

[1] Hurley, S., Consciousness in action, Cambridge, MA: Harvard University Press, 1998.

[2] Brooks, R., "New approaches to robotics". Science, 253, pp. 1227–1232, 1991.

[3] Krüger, V., Herzog, D., "Tracking in object action space," Computer Vision and Image Understanding, in press.

[4] Shanahan, M. P., "Embodiment and the Inner Life," Cognition and Consciousness in the Space of Possible Minds. Oxford: Oxford University Press, 2010.

[5] Ziemke, T., "The construction of reality in the robot: Constructivist perspective on situated artificial intelligence and adaptive robotics," Foundations of Science, 6, pp. 163–233, 2001.

[6] Piaget, J., The psychology of intelligence, London: Routledge and Kegan Paul, 1951.

[7] Varela, F., Thompson, E. and Rosch, E., The embodied mind: Cognitive science and human experience. Cambridge: MIT Press, 1991.

[8] Froese, T. and Ziemke, T. (2009). Enactive artificial intelligence: Investigating the systemic organization of life and mind. Artificial Intelligence, 173(3-4), 466–500.

[9] O'Regan, K., Noë, A., "A sensorimotor account of vision and visual consciousness," Behavioral and Brain Sciences, 24(5), pp 939–73, 2001.

[10] Steels, L. (2004). The Autotelic Principle. In I. Fumiya, R. Pfeifer, L. Steels, & K. Kunyoshi (Eds), Embodied Artificial Intelligence (pp. 231-242), Springer Verlag.

[11] Georgeon, O., Marshall, J., and Gay, S., "Interactional motivation in artificial systems: between extrinsic and intrinsic motivation," 2nd International Conference on Development and Learning and on Epigenetic Robotics (EPIROB2012), San Diego, pp. 1-2, 2012.

[12] Kaelbling, L., Littman, M., and Cassandra, A., "Planning and acting in partially observable stochastic domains". Artificial Intelligence Journal, 101, pp.99-134, 1998.

[13] Pfeifer, R. and Scheier, C., "From perception to action: The right direction?" In P. Gaussier and J.-D. Nicoud (Eds.), From Perception to Action (pp. 1-11). IEEE Computer Society Press, 1994.

[14] Oudeyer, P.-Y., Kaplan, F. and Hafner, V., "Intrinsic motivation systems for autonomous mental development," IEEE Transactions on Evolutionary Computation, 11 (2), pp. 265-286, 2007.

[15] Sun, R. and Giles, C. L., Sequence Learning - Paradigms, Algorithms, and Applications, Berlin Heidelberg: Springer, 2000.

[16] Georgeon, O., and Ritter, F., "An intrinsically-motivated schema mechanism to model and simulate emergent cognition," Cognitive Systems Research, 15-16, pp. 73-92, 2012.

[17] Georgeon, O. and Marshall, J., "Demonstrating sensemaking emergence in artificial agents: A method and an example," International Journal of Machine Consciousness, in press.

[18] Mondada, F. et al., "The e-puck, a robot designed for education in engineering," 9th Conference on Autonomous Robot Systems and Competitions, pp. 59-65, 2009.

[19] McCallum, A., "Learning to use selective attention and short-term memory in sequential tasks". Fourth International Conference on Simulating Adaptive Behavior; 1996.

[20] Drescher, G. L., Made-up minds, a constructivist approach to artificial intelligence. Cambridge, MA: MIT Press, 1991.