2015

Modeling Sensorimotor Learning in LIDA Using a Dynamic Learning Rate

D. Dong

S. Franklin

Recommended Citation
Dong, D., & Franklin, S. (2015). Modeling Sensorimotor Learning in LIDA Using a Dynamic Learning Rate. 10.1016/j.bica.2015.09.005

Modeling Sensorimotor Learning in LIDA Using a Dynamic Learning Rate

Daqi Dong and Stan Franklin
The University of Memphis, TN, U.S.A.
ddong@memphis.edu, franklin@memphis.edu

Abstract
We present a new model of sensorimotor learning in a systems-level cognitive model, LIDA. Sensorimotor learning helps an agent properly interact with its environment using past experiences. This new model stores and updates the rewards of pairs of data, motor commands and their contexts, using the concept of reinforcement learning; thus the agent is able to generate (output) effective commands in certain contexts based on its reward history. Following Global Workspace Theory, the primary basis of LIDA, the process of updating rewards in sensorimotor learning is cued by the agent's conscious content—the most salient portion of the agent's understanding of the current situation. Furthermore, we added a dynamic learning rate to control the extent to which a newly arriving reward may affect the reward update. This learning rate control mechanism is inspired by a hypothesis from neuroscience regarding memory of errors. Our experimental results show that sensorimotor learning using a dynamic learning rate improves performance in a simulated movement of pushing a box.

Keywords: Sensorimotor Learning, action execution, LIDA model, cognitive modeling, Memory of errors

1 Motivations and Background Introduction

We present two contributions in this paper. First, we implement a model of sensorimotor learning in LIDA[*] (Franklin, Madl et al. 2014) using the concept of reinforcement learning; this work provides LIDA with new capabilities. Second, we introduce the effect of memory of errors (Herzfeld, Vaswani et al. 2014) into this learning mechanism, so as to implement a dynamic learning rate.

The LIDA Model is a conceptual, systems-level model of human cognitive processes, used to develop biologically-inspired intelligent software agents and robots. It is primarily based on Global Workspace Theory (GWT) (Baars 1988, Baars 2002), and implements a number of additional psychological and neuropsychological theories. The LIDA model is grounded in the LIDA cognitive cycle (see Figure 1). Each cognitive cycle consists of three phases: 1) an agent (Franklin and Graesser 1997) first senses the environment, recognizes objects, and updates its understanding of the current

[*] For historical reasons LIDA stands for Learning Intelligent Distribution Agent.

situation in its Current Situational Model; 2) then in the Global Workspace it decides by a competitive process what portion of the represented situation should be attended to as the agent's conscious content, and broadcasts this portion to the rest of the system in order to facilitate action selection and modulate learning; 3) finally, the agent learns into its various memory systems, and simultaneously chooses an appropriate action, and executes it.

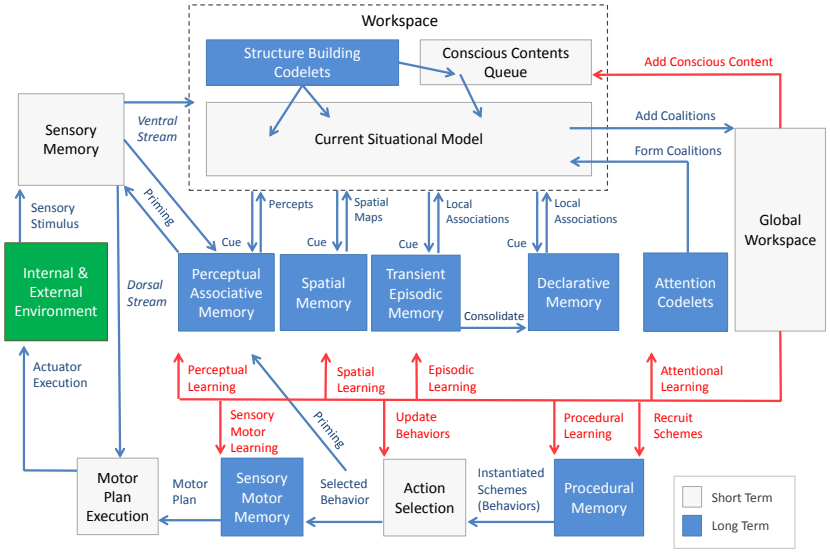

Figure 1: LIDA Cognitive Cycle Diagram

We have recently implemented a model for action execution known as the Sensory Motor System (SMS) (Dong and Franklin 2014). Action execution refers to a process by which an agent or robot executes a selected goal-directed action so as to output pertinent movement. The SMS generates executable motor commands driven by both the goal-directed action and low-level sensory data, and sends them to the agent's actuators. The SMS is the key module that is augmented by our implementing a model of sensorimotor learning into LIDA.

The SMS is implemented in LIDA using two modules: Sensory Motor Memory and Motor Plan Execution as depicted in the bottom left corner of Figure 1. In the SMS, the mechanism of motor command generation is designed according to the concept of the subsumption architecture (Brooks 1986, Brooks 1991), a type of reactive motor control mechanism. Inside the subsumption architecture, a sensor is directly linked to the motor that drives the actuators, and there is no central control (Connell 1989, Brooks 1990). This feature of no central control matches required features of action execution in the SMS. As mentioned by Marc Jeannerod, "…the conceptual content, when it exists (i.e., when an explicit desire to perform the action is formed), is present first. Then, at the time of execution, a different mechanism comes into play where the representation loses its explicit character and runs automatically to reach the desired goal…" (2006). This capability of running automatically to reach the desired goal without "its explicit character" is accomplished computationally by using the subsumption architecture in the SMS. However, in LIDA, the SMS is a module that cooperates with other LIDA modules, some of which maintain representations of environment internally, contrary to a commitment of the subsumption architecture, for example, some modules occurring in the understanding and attention phases of the LIDA cognitive cycle.

In the next section, we introduce the work of modeling sensorimotor learning in LIDA. Then in Section 3, we describe an addition of a dynamic learning rate into this learning mechanism. Following that, we provide current experimental results. Finally, we propose some directions for further research.

2 Modeling Sensorimotor Learning in LIDA

Practically it is easy to study this modeling work by using an example that includes concrete motor commands. Therefore, we simulated an autonomous agent (Franklin and Graesser 1997) to implement our model of sensorimotor learning. The agent consists of a simulated robot body and a controller implemented using a computational LIDA framework (Snaider, McCall et al. 2011). This agent is designed to learn how to push a box properly.

Below we introduce the robot, the cooperation between the Sensory Motor System (SMS) and some of LIDA's other modules, the development of the new SMS, and the implementation of other relevant LIDA modules.

2.1 A Box Pushing Robot

We reuse a two-wheeled robot provided by Webots[†], as shown in Figure 2. Its simulated height is 0.08m and its radius is 0.045m. The motor commands of the robot are limited to going forward, and turning left or right either by approximately 13 or 22 degrees.

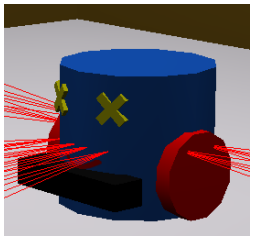

Figure 2: A two-wheeled box pushing robot

The robot has eight distance sensors. Each of these sensors has a field of view of about 15 degrees, and can detect objects within 0.3m. We simplified the sensors to detect objects in two distance ranges, one from 0m to 0.15m (NEAR) and the other from 0.15m to 0.25m (FAR). These sensors are arranged in an orthogonal pattern, with four on the front and two on each side. In addition, a touch sensor is placed on the robot's front to detect a bump, and another sensor built inside the robot's body detects when the robot becomes stuck—the sensor senses the agent's location and rotation; it is activated when the agent's location and rotation are the same during two consecutive sensory cycles.

2.2 The Cooperation between the SMS and Some Other LIDA Modules

Before modeling sensorimotor learning, there are two LIDA modules, Action Selection and Sensory Memory (see Figure 1), that provide relevant information—a selected behavior and the sensory data through a dorsal stream channel[‡] respectively—as separate inputs to the SMS. The SMS sends out motor commands as its output to the Environment module.

We implemented a broadcasting channel from LIDA's Global Workspace (GW) to the SMS (Sensory Motor Memory in Figure 1), sending the agent's conscious content to the SMS. The arrival of this content cues (initializes) the update of the rewards to motor commands in the SMS so as to assign credits to effective commands. Note that the conscious content does not directly provide the rewards, but it leads to the process of making and then updating the rewards. This is in keeping with GWT, in which the conscious content broadcast from the GW modulates learning in the rest of the system.

[†] Webots is a mobile robot simulation software package (www.cyberbotics.com).

[‡] In LIDA, a dorsal stream channel directly passes sensory data from the sensory memory to the action execution process.

2.3 The Development of the SMS

The SMS is the key module that was augmented when we implemented a model of sensorimotor learning into LIDA. Prior this addition, motor commands were built into a mechanism (subsumption architecture style) implemented in the SMS and could not be changed at runtime. Now, the sensorimotor learning implemented into LIDA leads to a dynamic selection of motor commands at runtime based on the newly experienced rewards to the commands. Our computational design is inspired by Mahadevan and Connell's previous work (1992). They added a learning aspect into the traditional subsumption architecture using the idea of reinforcement learning (Kaelbling, Littman et al. 1996). We improved theirs in two primary ways: 1) we imbued the original learning with more biologically inspired interpretation by bringing it into LIDA to implement sensorimotor learning, and 2) we implemented a new mechanism to control the rate of learning (see this in Section 3 later).

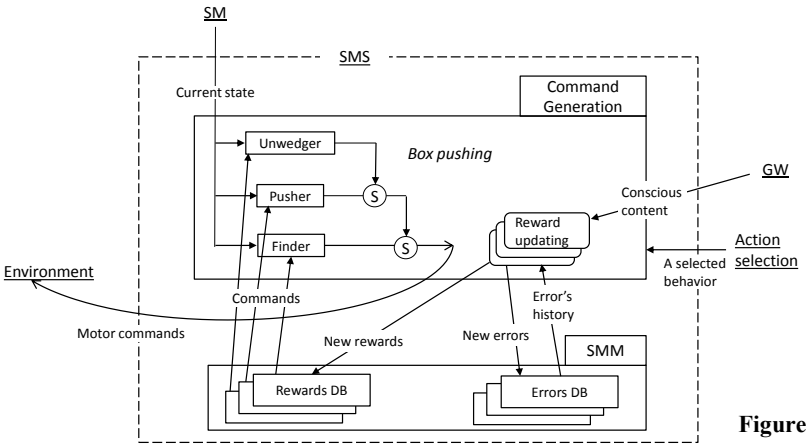

Figure 3: The design of a new SMS

The design of the new SMS is shown in Figure 3. Compared to its previous version, we have added three new components: (1) a set of reward updating processes, (2) a set of rewards databases, and (3) a set of errors databases. We introduce the first two components and the SMS for modeling sensorimotor learning below, and leave the introduction of the errors databases for adding a dynamic learning rate to Section 3.

In Figure 3, the SMS contains a (motor) command generation module depicted in the upper part of the diagram and a long-term memory, Sensory Motor Memory (SMM), depicted in the bottom part. The command generation module responds to the execution of a selected behavior. That behavior results from the preceding Action Selection Module on the right, and acts to specify a desired action in LIDA. General details about the behavior data structure can be found in (Franklin, Madl et al. 2014). In our case, the selected behavior is pushing a box. On the left side of this module, a reactive motor control mechanism—a kind of subsumption architecture—is built in. The structure of the mechanism implements a priority network that imposes an order on the three sub-tasks of box pushing. The unwedger's behavior suppresses the pusher's, and both of these suppress the finder's. A suppression operation is represented by an encircled uppercase S in the network diagram. Briefly, the agent begins by finding and closing a box, it then continuously pushes the box, and finally the agent can "unwedge" itself if it becomes stuck. These sub tasks are implemented by finite state machines (FSMs), which are driven by the current state of the environment sensed through the Sensory Memory (SM), and which may in certain states send out motor commands to the Environment module.

Because of the implemented sensorimotor learning, the motor commands sent out from the FSM can now be dynamically chosen at runtime based on their rewards. Each FSM has its own rewards database maintained in SMM. Another part of learning is a set of reward updating processes, which

are depicted on the right side of the command generation module. These processes are driven by the conscious content broadcast from LIDA's Global Workspace (GW), and each of them one-to-one updates the reward values stored in a rewards database for a certain FSM.

The algorithm of the reward updating process is inspired by Q-Learning (Watkins 1989); this updating helps the agent propagate rewards in the time line. The reward update formula (see Eq. (1)) uses a reward function Q(x, m) across states (x) and motor commands (m). This reward function is defined by Q(x, m) = r + ϒE(y), where *r* is the immediate reward, and E(y) is the expected reward of the state y resulting from the command. E(y) is the maximum Q(y, m) over all commands m. *ϒ* is a discount parameter that is set to 0.9, which determines the importance of future rewards. Its current value 0.9 is supported by Mahadevan and Connell's experimental results (1992).

$$Q(x, m) \leftarrow Q(x, m) + \beta(r + \Upsilon E(y) - Q(x, m)) \quad (1)$$

During updating, since the current stored reward Q(y, m) has not yet converged to the updated value—r + ϒE(y)—the difference between them then provides the reward error in the current stored rewards. This error is used to update the stored rewards using a learning rate *β*. Currently, the value of *β* is set to 0.5 as supported by Mahadevan and Connell's experimental results (1992); but we will replace it using a dynamic learning rate mechanism described in Section 3 later.

In Eq. (1), immediate rewards (*r*) are calculated differently depending on the FSMs' behavior (see Figure 3). First, for finding a box, the agent is rewarded by +3 if it detects an object in its front NEAR zone during forward movement, or it is punished by -1 if no object is there. The default reward is 0 if the agent is not moving forward. Second, for pushing a box, the agent is rewarded by +1 if it is touching an object during its forward motion, or it is punished by -3 if not touching. The default reward is 0 as before. Finally, for getting unwedged, the agent is punished by -3 if it is wedged while moving forward, or it is rewarded by +1 if no wedging occurs. The default reward is 0 if the agent is neither wedged nor moving forward.

When a FSM chooses its current command, in 90% of the time, given the same current state, the motor command that has maximum reward values is chosen. In the remaining 10% of cases, a motor command is randomly chosen. Choosing commands only based on their rewards will never allow the exploration of new commands or new states. Sometimes, a random command is chosen to ensure that all states in the state space will eventually be explored. Suggested by Mahadevan and Connell (1992), 10% is a good compromise between exploratory and goal-directed activity, in line with their experimental results.

2.4 Implementation of Other Relevant LIDA Modules

We implemented several other relevant LIDA modules appropriate for the specification of learning a box pushing task using sensorimotor learning. We list the implementation of each module below, ordered according to the three phases of the LIDA cognitive cycle: understanding, attention, and action/learning. We describe them in a linear way in order to make the process more easily understood. But actually, each LIDA module acts independently and asynchronously with other modules in LIDA. Figure 1 gives an intuitive feel for the relationship of these modules. Details of these modules can be found in (Franklin, Madl et al. 2014).

The Environment Module The environment constantly 1) retrieves the data detected by the robot's sensors, and 2) sends out the motor commands provided by the SMS to the robot's actuators. See Section 2.1 for the detailed sensors and actuators in our case.

Sensory Memory (SM) SM gets sensory data from the Environment Module, structured as an array of Boolean bits, representing the data status sensed from each of the sensors, either active or inactive. SM provides the SMS with the current data.

Feature detectors (FDs) and Perceptual Associate Memory (PAM) PAM stores a set of nodes, each of them representing a specific aspect of an environmental state of concern to the agent. In our work, these nodes are distance nodes including NEAR and FAR, a bumping node, and a stuck node. FDs constantly obtain the current state from the SM, activating relevant nodes in PAM.

The Current Situational Model (CSM) and structure building codelets (SBCs) The CSM receives currently activated nodes from PAM, and builds the agent's understanding of the current situation. SBCs reorganize data in the CSM, combining sets of nodes and links into node/link structures. They build an agent's higher level understanding of the current situation.

Attention codelets and the Global Workspace (GW) We added an attention codelet concerned for the entire current situation in the CSM, and bringing it into the GW. In the GW, the current situation may win the competition to produce the agent's conscious content. A channel from the GW to the SMS—newly implemented this time (see Section 2.2)—broadcasts the conscious content to other modules including the SMS.

Procedural Memory (PM) Following the broadcast of conscious content from the GW, a box pushing scheme is recruited in the PM, and then a relevant behavior is instantiated that is selected by the Action Selection module and sent to SMS, which initiates a motor command generation mechanism for executing the box pushing in the SMS.

3 A Dynamic Learning Rate in Sensorimotor Learning

In a study of sensorimotor learning, Herzfeld and colleagues (2014) have hypothesized that the brain not only learns from individual errors that it has experienced before, but also accumulates the errors into a memory; this memory of errors makes it possible for the brain to control how much it will learn from a given current error. These historical errors help humans determine whether the environment is steady or rapidly changing. Environmental stability thus controls how much of a given error will be learned so as to affect the prediction of the upcoming movement.

We interpret the above hypothesis as a computational mechanism. In the mechanism, memory of errors controls the value of a "learning rate", which weights the amount of an error—typically the difference between sensory (actual) result and predicted (intended) result—that will be used in an update process of the predicted result.

In the work of modeling sensorimotor learning as described above in Section 2, one step in updating the reward of motor commands is to learn from the reward error as expressed by Eq. (1). We consider this reward updating process to be similar to the update process mentioned in the above mechanism. Therefore, here we revise Eq. (1) by changing the quality of its parameter β from a constant value to a dynamic value. Now the value of parameter β is controlled by memory of errors as introduced above (Herzfeld, Vaswani et al. 2014).

Note that the reward of motor commands manipulated in our development of sensorimotor learning is different than the execution result (the movement) of motor commands discussed by those neuroscience researchers in sensorimotor learning (Herzfeld, Vaswani et al. 2014). Reward of motor commands is maintained inside an agent as the judgment provided by the agent's mind. In contrast, the movement of motor commands occurs outside of the agent as the feedback "provided" by the environment. We consider the reward and the movement to be the two indicators for the evaluation of a motor command. We note that both of them are used to indicate aspects of motor commands, and it seems they also have some similar principles, such as the way to update their indications of motor commands—they both use a type of learning rate to weight the updating of the old knowledge of the motor command by the new. Thus, we have integrated the idea of memory of errors from Herzfeld and

colleagues (2014) into our work for updating the reward of motor commands. We consider this a kind of indirect biological inspiration for our approach.

Next we provide the way in which memory of errors dynamically controls the value of the parameter β. First, a (reward) error is classified between two types, either positive or negative, depending on the sign of subtracting the old (stored) reward from the new reward. Then when our agent has performed its task (pushing box) for a while, having experienced a sequence of errors, the type of these errors may switch differently, switching from slowly to rapidly; thus we have different types of memory of errors from the viewpoint of their stability, based on the rate of switching. We present the number of errors the agent has experienced by a variable n, and the number of switches between these errors by a variable s. These variables are integers starting from zero. Based on these two variables, we represent the status of the memory of errors by a variable t as expressed by Eq. (2), and then calculate the value of β using a sigmoid function assisted by the variable t as expressed by Eq. (3). The parameter θ tunes the effect of t, and is set to 1.0 by default. β ranges from 0.0 to 1.0 with a default value of 0.5.

$$t = n - 2 * s \quad (2)$$

$$\beta = \frac{1}{1 + e^{-t*\theta}} \quad (3)$$

Here we illustrate the behavior of β with examples. If the agent has experienced many errors that rarely switch, the value of n is large and the value of s is small; therefore, the value of t and therefore θt are large; so β is close to 1.0. This means that a slowly switching environment results in a high learning rate. On the other hand, if there are errors in memory but they have switched signs very often, the values of both s and n are large, so t is negative with a large absolute value; thus the value of β is close to 0.0. This means a rapidly switching environment leads to a low learning rate. These simulated behaviors qualitatively agree with the hypothesis proposed by Herzfeld et al. (2014). Note that when there is no error in the memory yet, the value of t is 0 because n and s are 0, so the value of β is 0.5, which is the same as the default value of β.

Finally, we add error databases to store both variable n and variable s. These databases are maintained in the SMM as shown in Figure 3. They interact with reward processes to (1) update memory of errors based on the arrival of new errors, and (2) provide current error's history to dynamically control the value of β.

4 Experimental Results

This section describes an experimental study to evaluate this modeled sensorimotor learning and its dynamic learning rate. Figure 4 shows a bird's eye view of the experimental environment and the agent in their initial configuration. The agent stands in a field containing three movable boxes. They are surrounded by walls, a kind of obstacle that cannot be moved.

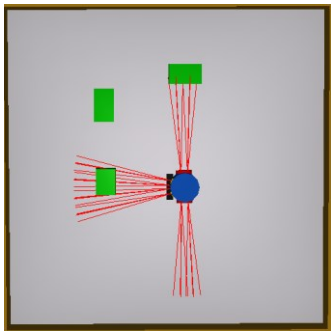

Figure 4: A bird's eye view of the experimental environment and the agent

We are interested in the following questions: 1) How well will the action of pushing a box be executed with sensorimotor learning; and 2) What is the effect of implementing the dynamic learning rate into the learning?

We evaluate the performance of the box pushing using two criteria: 1) the average value of the reward obtained so far by the pusher FSM (see Figure 2) inside the agent—we consider pushing to be the core part of the box pushing task—and 2) the distance that the boxes have been moved in the environment.

We compare the box pushing performance across six different agent conditions: 1) randomly chosen motor commands; 2) handcoded motor commands; 3-5) sensorimotor learning with constant learning rates of 0.1, 0.5, and 0.9 respectively; and 6) learning with a dynamic learning rate. Under the handcoded condition, the expected "best" commands are chosen when the agent is in certain given states: the finder module chooses forward motion if an object has been detected to be near the front; the pusher chooses forward if a bump event is detected; and the unwedger module randomly chooses a command to turn left or right if the agent is stuck. In other states, commands are randomly chosen.

In each condition, we perform 10 consecutive trials. A new trial begins from the initial configuration of the environment and the agent, but the rewards of the motor commands and the reward errors are remembered throughout the trials. During each trial the agent runs 500 simulation steps, so under each condition, the agent runs 5000 steps. In our case, each step is simulated with 50 virtual time units—the agent executes at unit intervals in Webots' virtual time.

We collected the first criterion of average value of the reward every 50 steps of the agent's run, so 100 average values were collected during the total of 5000 steps. Figure 5 plots the average values under the six conditions. The vertical axis represents the sum of all rewards obtained by the pusher divided by the number of steps the agent has run so far, and the horizontal axis represents the completed proportion of the 5000 steps. The plotted curves illustrate that 1) with sensorimotor learning added, the pusher module obtains more rewards than the random agent does, 2) the pusher obtains the most rewards under the dynamic learning rate condition (except during the initial steps), and 3) the handcoded agent outperforms the agents with the three constant learning rates.

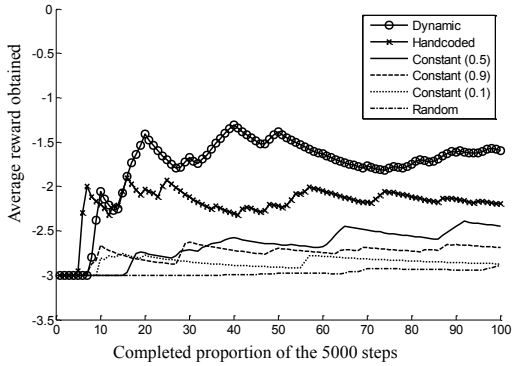

Figure 5: The average values of rewards obtained by the pusher

Table 1: The sums of the distances that the boxes have been moved during 10 trials

Learning Rate	Distance (m)
Dynamic	1.6002
Handcoded	1.2023
Constant (0.5)	0.9521
Constant (0.9)	0.6357
Constant (0.1)	0.3223
Random	0.2338

Regarding the second criterion, the sums of the distances that the boxes have been moved during 10 trials under each of the six conditions are displayed in Table 1. These show that 1) the agents with sensorimotor learning at the three constant rates have pushed the boxes farther than the random agent, 2) using the dynamic learning rate yields the greatest box pushing distance, and 3) the handcoded agent yields the second greatest distance.

The results reported above support the assertion that sensorimotor learning improves the performance of box pushing to a certain extent, and that adding the dynamic learning rate clearly increases that extent. This increased improvement supports that memory of errors—the agent's

knowledge of the environment's stability—helps the agent interact with its environment more effectively.

We did not compare our results with the results obtained by Mahadevan and Connell (1992) because they have different experimental motivations, and thus a different types of results. They are interested in determining 1) the effect of decomposing the overall task into a set of subsumption modules for learning, and 2) the performances of different learning algorithms, while we are interested in the biologically inspired implementation of sensorimotor learning, and adding a learning rate control mechanism inspired by the idea of memory of errors (Herzfeld, Vaswani et al. 2014).

5 Summary and Future work

In this paper, we implemented sensorimotor learning in LIDA (Franklin, Madl et al. 2014). This implementation follows the ideas of Global Workspace Theory, and uses reinforcement learning. Furthermore, we added a dynamic learning rate into the learning, which is controlled by history of errors. Our preliminary experimental results suggest that sensorimotor learning using a dynamic learning rate improves the performance of action execution. On the other hand, we think more evidence is needed to support the causality between using a dynamic learning rate and the learning performance. Our motivation for modeling this dynamic learning rate is the replication of some recently proposed hypotheses from neuroscience regarding the effect of memory of errors (Herzfeld, Vaswani et al. 2014). In brief, the hypotheses suggest that a dynamic learning rate helps an agent achieve a better adaptation to its environment based on its memory of errors, but can that adaptation always be translated into improved performance? We leave this as an issue for future work.

References

Baars, B. J. (1988). A cognitive theory of consciousness. New York, Cambridge University Press.

Baars, B. J. (2002). "The conscious access hypothesis: origins and recent evidence." Trends in cognitive sciences **6**(1): 47-52.

Brooks, R. A. (1986). "A robust layered control system for a mobile robot." IEEE Journal of Robotics and Automation **2**(1): 14-23.

Brooks, R. A. (1990). "Elephants don't play chess." Robotics and autonomous systems **6**(1): 3-15.

Brooks, R. A. (1991). "How to build complete creatures rather than isolated cognitive simulators." Architectures for Intelligence: The Twenty-second Carnegie Mellon Symposium on Cognition: 225-239.

Connell, J. H. (1989). A colony architecture for an artificial creature, DTIC Document.

Dong, D. and S. Franklin (2014). Sensory Motor System: Modeling the process of action execution. Proceedings of the 36th Annual Conference of the Cognitive Science Society, Austin TX, USA.

Franklin, S. and A. Graesser (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Intelligent agents III agent theories, architectures, and languages. London, UK, Springer-Verlag**:** 21-35.

Franklin, S., et al. (2014). "LIDA: A Systems-level Architecture for Cognition, Emotion, and Learning." IEEE Transactions on Autonomous Mental Development **6**(1): 19-41.

Herzfeld, D. J., et al. (2014). "A memory of errors in sensorimotor learning." Science **345**(6202): 1349-1353.

Jeannerod, M. (2006). Motor cognition: What actions tell the self. Oxford, UK, Oxford University Press.

Kaelbling, L. P., et al. (1996). "Reinforcement learning: A survey." arXiv preprint cs/9605103.

Mahadevan, S. and J. Connell (1992). "Automatic programming of behavior-based robots using reinforcement learning." Artificial intelligence **55**(2): 311-365.

Snaider, J., et al. (2011). The LIDA framework as a general tool for AGI. Artificial General Intelligence. Berlin Heidelberg, Springer 133-142.
Watkins, C. J. C. H. (1989). Learning from delayed rewards, University of Cambridge.