Simulation of a Motivated Learning Agent

Janusz A. Starzyk[1], James Graham[1], Leszek Puzio[2]

[1]Ohio University, Athens, OH 45701 USA
{starzykj, jg193404}@ohio.edu
[2] WSIZ, Rzeszow, 35-225 Poland
puzio@wsiz.rzeszow.pl

Abstract. In this paper we present our work directed at building a simple motivated learning agent with symbolic I/O. To do this we created a simulation environment within the NeoAxis game engine. The purpose of this work is to explore autonomous development of motivations and memory in agents within a simulated environment. The approach we took should speed up the development process, bypassing the need to create a physical embodied agent as well as reducing the learning effort. By rendering low-level motor actions such as grasping or walking into symbolic commands we remove the need to learn elementary motions. Instead, we have several basic primitive motor procedures in a procedural memory, which can form more complex procedures. Furthermore, by simulating the agent's environment, we both improve and simplify our control over the learning process. As a result, there are fewer adaptive learning variables associated with both the agent and its environment, and learning takes less time, than it would in a more complex real world environment.

Keywords: Motivated learning, cognitive architectures, simulation, embodied intelligence

1 Introduction

A significant challenge in robotics is to develop autonomous systems that can reason and perform missions in dynamic, uncertain, and uncontrolled environments [1]. Therefore, recent research efforts are directed towards developing autonomous cognitive systems. Existing methods have made a significant progress in this direction [2,3,4,5] and the topic is actively researched in laboratories around the world.

Current cognitive architectures, such as SOAR [6], ACT-R [7], Icarus [8], LIDA [9], Polyscheme [10], and CLARION [11], either have to rely on predefined goals (without self-motivated learning) or predefined rules (without autonomous reasoning). Due to their reliance on predefined scripts and heuristic rules, current robotic systems lack autonomy, self-adaptability, and reasoning capabilities either to accomplish complex missions or to handle ever changing missions in uncontrolled environments.

Another important direction in studying development of cognitive systems and robots is based on the idea of embodied intelligence. The principles of designing robots based on the embodied intelligence idea were first described by Brooks [12] and were

characterized through several assumptions that would facilitate development of embodied agents.

Since our aim is to develop intelligent machines we introduce internal motivations, creating abstract goals not previously known to the designer or the robot. Intelligent systems will adapt to unpredictable and dynamic situations in the environment by learning, which will give them a high degree of autonomy, making them a perfect choice for robotics and virtual agents [13]. The recently developed mechanism of motivated learning (ML) has such capacities [14].

With ML, robots can achieve various goals imposed by different challenge scenarios autonomously. They develop higher level abstract goals and increase internal complexity of representations and skills stored in their memory. Our aim in this work is to develop simulation tools of virtual autonomous systems with ML mechanism.

Most current autonomous robot systems concentrate on the cognitive development of individual robots [15,16,17]. They mainly focus on developing simple local behavior control algorithms under heuristic rules, and then seek to emerge global behaviors. Adding intrinsic motivations and advanced reasoning capabilities improve the robots' individual capabilities. In addition, improving robots' learning in complex dynamically changing environments is very important.

Therefore, we work to provide a systematic framework for developing cognitive robots that can autonomously accomplish a wide variety of real-world complex missions in dynamic, uncertain environments. We have selected NeoAxis to build a virtual 3D environment for embodied motivated agents. That environment is able to simulate wide scope of robot types, ranging from wheeled robots, along with flying or swimming robots, to humanoid robots. The second reason why we utilize NeoAxis is that NeoAxis has good support for physics modeling. We can assign static and dynamic friction parameters, mass, bounciness, hardness, etc., to obtain real-world representation of objects and different material types. Objects could be attached to one another to create complex structures, like a car is composed of wheels, body, windscreen, and engine. It is also possible to create environment rules, i.e., a tree produces apples in certain intervals or times.

The rest of this paper is organized as follows: In the section 2, we discuss the motivated learning agent and how it learns to interact with its environment. We discuss how pains are generated and adjusted and how goals are selected. Following this in section 3, we discuss the simulation of a virtual OML agent. Finally, in section 4, we discuss how we integrated the agent into the NeoAxis environment. This includes our current work, and our plans to further advance the simulation tool.

2 Motivated Learning Agent Memory Organization

The motivated learning (ML) agent interacts with the virtual environment changing it by its actions and receiving rewards (external and internal) for its actions. In this implementation of the motivated learning agent we assume that both sensory inputs and motor outputs are symbolic, and they provide interface to the virtual environment.

ML uses a neural network where each sensory neuron represents an object and each motor neuron represents an action.

The ML system's neural network, in addition to sensory S and motor M neurons, contains pain center neurons P that register the pain signals, and goal neurons G responsible for pain reduction. Selected pain center neurons are connected to the external reward/punishment signals. In RL these neurons receive a reward or punishment signal according to the training algorithm, and in ML they receive primitive pain signals that directly increase or decrease their activation level. In ML, abstract pain centers are created through the goal creation mechanism [14,18] and are activated via an interpretation of sensory inputs. A goal is an intended action that involves a sensory-motor pair. To implement a goal the agent acts on the observed object. All pain neurons are initially connected to goal neurons with random interconnection weights. All goal neurons and pain neurons are subject to Winner-Take-All (WTA) competition between them. The number of goal neurons is equal to the number of sensory-motor pairs. In the symbolic representation each neuron represents a single symbol, pain, goal or action. Fig. 1 shows symbolically the interconnection structure, between S, P, B, G and M neurons. In Fig. 1 an abstract pain center P_k connections to its sensory, bias, goal and motor neurons are shown.

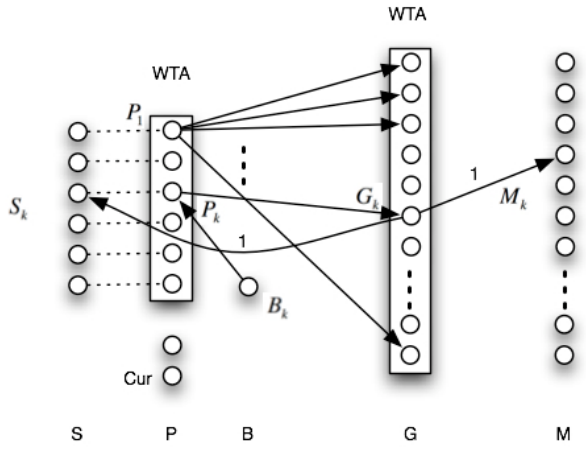

Fig. 1. Connections between sensory, motor, bias, pain and goal neurons.

2.1 Bias signals, weights, and associated pains

A bias signal triggers an abstract pain and is defined depending on the type of perceived situation. If the autonomous agent needs to maintain a certain level of resources, the bias reflects how difficult it is to obtain this resource or in a more general case, how difficult it is to perform a desired action. Resources can be either desired if their use can reduce the agent's pain or undesired if they can increase the pain. Thus the agent must first have an experience to determine if the resource is desired or undesired to introduce a resource related bias signal.

The bias signal for desired or undesired resources is calculated from the resource level of resource and its desired/undesired limits as follows:

$$B = \gamma * \left(\frac{\varepsilon + R_d(s_i)}{\varepsilon + R_c(s_i)}\right)^{\delta_r} \quad (1)$$

where R_d is a desired resource value (observed at a sensory input s_i). ε is a small positive number to prevent numerical overflow, γ regulates how quickly pain increases, $\gamma > 0$ and δ_r =1 when the resource is desired, δ_r =-1 when it is not desired, and $\delta_r = 0$ otherwise (when the character of the resource is unknown).

Initially all B-P_k weights w_{bp} are set to 0. Thus, the machine initially responds only to the primitive pain signals P directly stimulated by the environment. Each time a specific pain P is reduced the weight w_{bp} of the B-P_k bias link increases. However, if the goal activated by the pain center P was completed and did not result in reduction of pain P, then the B-P_k weights w_{bp} are reduced. Since the bias weight B-P_k indicates how useful it is to have access to a desired S, a bias weight adjustment parameter Δ_b must be properly selected to reflect the rate of stimuli applied to a higher order pain center. This rate reflects how often a given abstract pain center P_k was used to reduce the lower order pain signal P.

2.2 Changes of the goal related and curiosity weights

Initial weights between P-G neurons are randomly selected in the 0-α_g interval (a good setting will be between 0.49 and 0.51 of α_g for faster learning). Assume that the weights are adjusted upwards or downwards by a maximum amount μ_g. In order to keep the interconnection weights within prespecified limits (0< w_{pg} <α_g), the value of the actual weight adjustment applied can be less than μ_g and is computed as

$$\Delta_a = \mu_g \ \min(|\ \alpha_g - w_{pg}\ |, w_{pg}) \qquad \text{where } \alpha_g \le 1 \quad (2)$$

and

$$\mu_g = \mu_0 \left(1 - \frac{2}{\pi}\text{atan}\left(10 * \left(\frac{R_c(s_i)}{R_d(s_i)}\right)^{\delta_r}\right)\right) \qquad \text{where } \mu_0 = 0.3 \quad (3)$$

Using (4) produces weights that slowly saturate towards 0 or α_g. (For quick learning set ($\mu_g = \alpha_g / 2$). No other weights from other pain centers to this specific goal are changed, so the sum of weights incoming to the node G is not constant.

If, as a result of the action taken, the pain that triggered this action increased (as determined by pain reduction parameter δ_p), then the w_{pg} weight is decreased by Δ_a, and if the pain decreased, then the w_{pg} weight is increased by Δ_a

$$w_{pg} = w_{pg} + \delta_p * \Delta_a. \quad (4)$$

2.3 Action value determination for OML agent:

In the opportunistic ML agent (OML) the "best action" is determined by the linear heuristic OML model, using action "Value" V_i

$$V_i = \frac{P_i + (\Delta P * (t_{mot} + t_{dist}))}{(t_{mot} + t_{dist})^2} \quad (5)$$

where ΔP is the estimated change in pain over 1 cycle for the primitives. P_i is the pain associated with the action under consideration, t_{mot} is the required motor time to complete the action, and t_{dist} is the time required by the agent to travel to a distant location to perform the action. We generally assume that pains won't chance over the course of the action. The action with the highest value of V_i is the one chosen by the OML agent. The selection of actions evaluated in (5) depends on the number of pains above threshold and whether or not they have been tried previously. For example, a known "good" action will have precedence over one of unknown utility even if the "unknown" action may be more advantageous in terms of distance to travel and potential pain reduced, simply because it is an unknown quantity.

3 NeoAxis Implementation

A cognitive architecture organization based on the ML idea was introduced in [19]. Since this architecture uses an emergent systems approach, it is essential to have a physical body and a physical environment. However, making a physical robot is expensive in terms of costs and design effort, so it is very helpful to use a computer simulation that can imitate real-time physical conditions. Thus the effort was switched from using a physical robot to making a simulated environment with a virtual robot.

We implemented the basic infrastructure of the ML agent in NeoAxis describing the motivated agent functionality in C++ and in C#. The virtual environment for ML agents built in NeoAxis is a 3D simulated world governed by realistic physics to present the robots with a complex, challenging world. This simulation environment can be separated into two major components. The first one is the animation controller that handles display tasks and transitions the agent from one action to another. The second component processes the agent's behaviors and defines the potentially sophisticated rules governing the virtual world in which the agent lives. The agent working in the created environment discovers these rules and learns to use them to its advantage.

To test the ML agent's learning process, we built a sample virtual environment. In this environment, we created resources that the agent could use (presented in Table 1), and we endowed the agent with the ability to act on the resources (listed in Table 1). The agent's actions are driven by pains. Only two pains listed in Table 1 as primitive pains are predefined. One is hunger which increases over time and the other is Curiosity. Curiosity pain makes the agent explore the environment when no other pain is detected and until valid actions are learned. Other pains are learned by the agent using the goal creation methodology. The agent observes which resources it needs and introduces the need to have them. We also defined world rules, which describes which agent actions make sense and what their results are. Those rules are listed in Table 1. The agent's actions result in various outcomes like increasing and decreasing resource quantities, as well as in reducing some pains.

Table 1. List of valid Resource-Motor pairs and their outcome

Motor	Resource	Outcome		
		Increase	Decrease	Pain reduce
Eat food from	Bowl		Food in Bowl	Hunger
Take food from	Bucket	Food in Bowl	Food in Bucket	Lack of food in Bowl
Buy food with	Money	Food in Bucket	Money	Lack of food in Stock
Work for money with	Hammer	Money	Hammer	Lack of Money
Study for job with	Book	Hammer	Book	Lack of Job
Play for joy with	Beach ball	Book	Beach ball	Lack of School

3.1 Simulation Algorithm of OML Agent in NeoAxis

In order to properly test our agent we needed to embed it in an environment and provide it with a means to observe the environment and interact with it. To do this, we expanded on our earlier testing methodologies and created the basis for a simple, but effective test bed within the NeoAxis engine. The basic steps of OML agent simulation in NeoAxis are as follows:

After initialization, the algorithm performs successive iterations. Each iteration consists of the Agent Phase, where the agent observes the Environment, updates its internal state, and generates motor outputs, and an Environment Phase where the environment performs the agent's actions and updates itself accordingly.

Our simulation of the virtual environment in NeoAxis implements all the preceding rules. To better visualize resources quantities, current task, pains levels and the agent's memory, we added windows to the simulation, which display the current state of the agent and the environment as shown in Fig. 2.

Fig. 2. Main simulation view with displayed simulation state in windows

When a pain level is above threshold it displays it in red. In this screenshot, the agent action is driven by 'Lack of Money' pain. The agent tries to learn valid actions and their outcomes. Sometime the agent takes a nonsense action like "Play for joy with hammer" in the simulation. Nevertheless, even actions such as this are useful for agent because by taking them, it learns that they are useless. The memory window, presented in Fig. 2 on the left, displays the memory state. When the color is gray then it means that the agent has not learned usefulness of this action yet. When the color is white then this means that the action is valid, if the color is black than the action is invalid. Each row in the "memory" corresponds to a driving pain, while each column represents a possible action. Once the agent learns all valid actions and its pains are under control, then the agent does the "Go rest on mattress" action. This motor action is a desirable final state for the agent.

We have run multiple simulations where we modified resources quantities and motor action times. By using a human controlled character we tried to disturb the ML agent via the by getting in its way or moving resources to different location. When starting resources were sparse, the agent couldn't learn all valid actions because it ran out of resources to test new actions. And sometimes when resources like food were plentiful, the agent did not bother to learn anything new once the hunger pain was under control. When action times were too long the agent couldn't satisfy all pains. But when we select proper simulation parameters the ML agent proves to learn correctly even in a complex and changing environment.

4 Conclusions

In this paper, we have presented our motivated learning agent with a focus toward simulating the agent within a graphical environment. We also included the discussion of several new modifications to our algorithm, including new calculations for bias signals and w_{pg} weights. Additionally, we introduced greater complexity into the environment by introducing undesirable resources. This includes changes in $\delta_p$calculation and the calculation of desired resource levels. By adding these new features we've improved the agent's ability to handle its environment as well as our own ability to implement complex and interesting environments for our agents to interact with.

The simulation results of the ML agent in the virtual 3D environment for embodied motivated agents in NeoAxis prove that our theoretical assumptions for motivated learning agent memory organization, determination of bias signals, weights, goal creation and selection, associated pain calculations, were valid. The OML agent was able to learn all environment rules, and keep the agent's pains under control.

Our further research will focus on the extension of the simulation, specifically, making a more complex environment and to introduce friendly and hostile characters. It will be also worthwhile to test if multiple ML agents could cooperate to obtain common goals.

References

1. Hirukawa, H., F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, K. Akachi, T. Kawasaki, S. Ota, K. Yokoyama, H. Handa, Y. Fukase, J. Maeda, Y. Nakamura, S. Tachi and H. Inoue. Humanoid robotics platforms developed in HRP. Robotics and Autonomous Systems 48:4, 165–175, 2004.
2. B. Bakker and J. Schmidhuber. Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization . In F. Groen, N. Amato, A. Bonarini, E. Yoshida, and B. Kröse (Eds.), Proceedings of the 8-th Conference on Intelligent Autonomous Systems, IAS-8, Amsterdam, The Netherlands, p. 438-445, 2004.
3. P-Y. Oudeyer et al. (2010). Intrinsically Motivated Exploration for Developmental and Active Sensorimotor Learning, in Sigaud, O. and Peters, J. eds., From Motor Learning to Interaction Learning in Robots, vol. 264/2010, Springer Berlin/Heidelberg, 107-146.
4. S. Ro et al. (2009). Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning, IEEE Intl. Conf. on Robotics and Biometrics, 665-670.
5. S. Singh, A.G. Barto, and N. Chentanez (2004). Intrinsically motivated learning of hierarchical collections of skills, Proc. 3rd Int. Conf. Development Learn., San Diego, CA, 112–119.
6. J.E. Laird, “Extending the Soar cognitive architecture,” in Artificial General Intelligence 2008, Memphis, TN: IOS Press, 2008, pp. 224-235.
7. J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, “An integrated theory of the mind,” Psych. Review, vol. 111, no. 4, pp. 1036–1060, 2004.
8. P. Langley and D. Choi, “A unified cognitive architecture for physical robots,” 21st Nat. Conf. Artificial Intelligence, Boston, MA: AAAI Press, 2006, pp. 1469-1474.
9. B.J. Baars and S. Franklin, “Consciousness is computational: the LIDA model of global workspace theory,” Int. J. Machine Consciousness. vol. 1, no. 1, pp. 23-32, 2009.
10. N. Cassimatis and L. Nicholas, Polyscheme: A Cognitive Architecture for Integrating Multiple Represetnation and Inference Schemes, MIT Ph.D. Diss., 2002.
11. R. Sun, “The importance of cognitive architectures: an analysis based on CLARION,” J. Experimental and Theor. Artif. Intell., vol. 19, no. 2, pp. 159-193, 2007.
12. R.A. Brooks “Intelligence without reason”, Proc. 12th Int. Conf. on Artificial Intelligence, pp. 569-595, Sydney, Australia, 1991.
13. R. Pfeifer, & J.C. Bongard. How the Body Shapes the Way We Think: A New View of Intelligence, The MIT Press (Bradford Books), 2007.
14. J. A. Starzyk, "Motivation in Embodied Intelligence" in Frontiers in Robotics, Automation and Control, I-Tech Education and Publishing, Austria, 2008, pp. 83-110.
15. A. Farinelli, L. Iocchi, and D. Nardi, “Multirobot systems: a classification focused on coordination,” IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 34, no. 5, pp. 2015-2028, 2004.
16. K. Lerman, C. Jones, A. Galstyan, and M.J. Mataric, “Analysis of dynamic task allocation in multi-robot systems,” Int. J. Robotics Research, vol. 25, pp. 225-241, 2006.
17. J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, “Autonomous mental development by robots and animals,” Science, vol. 291, no. 5504, pp. 599–600, Jan. 2001.
18. J. A. Starzyk, Motivated Learning for Computational Intelligence, in Computational Modeling and Simulation of Intellect: Current State and Future Perspectives, edited by B. Igelnik, IGI Publishing, ch.11, pp. 265-292, 2011.
19. J.A. Starzyk, “Mental Saccades in Control of Cognitive Process”, Int. Joint Conf. on Neural Networks, San Jose, CA, July 31 - August 5, 2011.