Simulation of a Motivated Learning Agent

Janusz A. Starzyk[1], James Graham[1], and 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 discuss how to design a simple motivated learning agent with symbolic I/O using a simulation environment within the NeoAxis game engine. The purpose of this work is to explore autonomous development of motivations and memory of agents in a virtual 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 use several basic primitive motor procedures, which can form more complex procedures. Furthermore, by simulating the agent's environment, we both improve and simplify the learning process. There are a few 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, virtual agents.

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 making a significant progress in this direction [2,3,4,5].

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.

H. Papadopoulos et al. (Eds.): AIAI 2013, IFIP AICT 412, pp. 205–214, 2013.
© IFIP International Federation for Information Processing 2013

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 improves the robots' individual capabilities. In addition, improving robots' learning in complex dynamically changing environments is very important, especially in environments where there may be competition for resources.

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 such environments. We have selected NeoAxis to build a virtual 3D environment for embodied motivated agents. That environment is able to simulate a wide scope of robot types, ranging from wheeled, flying or swimming robots, to humanoid robots. The second reason why we use NeoAxis is that it 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 composed of wheels, body, windscreen, and engine. It is also possible to create environment rules, i.e., a tree produces apples in certain time intervals.

The rest of this paper is organized as follows: In 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, and then we describe action value determination for opportunistic motivated learning (OML) agent. 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 we assume that both sensory inputs and motor outputs are symbolic, and provide interface to the virtual environment.

We assume that the ML agent learns in a hostile environment where either there are a limited amount of renewable resources that the agent needs or there are hostile characters that may harm the agent unless it learns to defend itself. A ML agent has a number of predefined basic needs e.g. desired energy level, comfortable temperature

zone, or acceptable pressure. The agent can satisfy these needs by taking proper actions. If a need is not satisfied (e.g. energy level is below threshold) the agent feels a pain signal. Thus, a pain signal related to basic needs can be predefined as a deviation from desired levels. If the agent learns how to satisfy its need then an abstract need is introduced in case the agent is unable to perform the desired action. Unsatisfied need manifests itself by a pain signal and the agent is motivated to reduce this pain.

An OML agent uses a neural network where each sensory neuron represents an object and each motor neuron represents an action. The OML system's neural network shown in Fig. 1, 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 the 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 also shown.

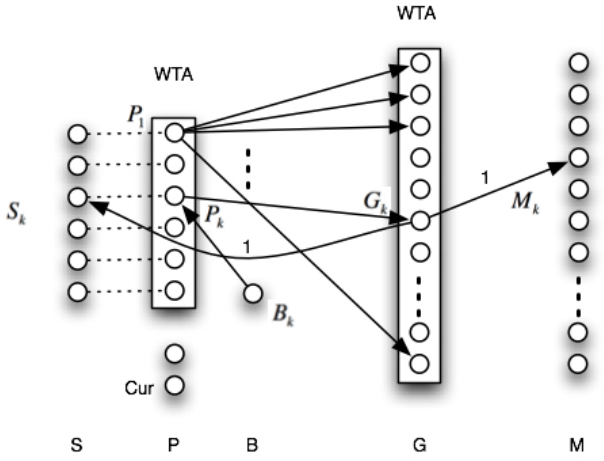

Fig. 1. Connections between sensory, motor, bias, pain and goal neurons

In what follows we explain how various weights of the OML system's neural network are adjusted during the learning process.

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 observed level of resource and its desired/undesired limits as follows:

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

where R_d is a desired resource value (observed at a sensory input s_i) and R_c is a current resource value. 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 resource S, a bias weight adjustment parameter Δ_b must be properly set 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.

When a specific goal is not invoked for a long period of time its importance in satisfying a lower level pain is gradually reduced. This requires a reduction of the w_{pg} weight to this goal from all the pain centers. A similar reduction of the B-P_k links indicates a gradual decline in importance of an abstract pain. This mechanism of lowering the weights to an abstract pain center prevents the machine from overestimating its importance to the lower level pain that was responsible for its creation.

Using the bias signal, the pain value is estimated from:

$$P(s_i) = B(s_i) * w_{bp}(s_i) \tag{2}$$

where w_{bp} is a bias to pain weight for a given pain center. w_{bp} is computed incrementally based on signals of pain change that resulted from the action taken as follows:

$$\begin{aligned} w_{bp} &= w_{bp} + \Delta_{b+}(\alpha_b - w_{bp}) && \text{if the associated pain was reduced or increased} \\ w_{bp} &= w_{bp}(1 - \Delta_{b+}) && \text{if there was no change in pain} \\ w_{bp} &= w_{bp}(1 - \Delta_{b-}) && \text{when the assoc. sensory input was not involved in this action} \end{aligned} \tag{3}$$

where α_b = 0.5, sets the ceiling for w_{bp}; Δ_{b-} = 0.0001, sets the rate of decline for w_{bp} weights; Δ_{b+} = 0.08 , sets the rate of increase for w_{bp} weights - increasing Δ_{b+} value increases the rate at which pains increase.

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

$$\Delta_a = \mu_g \ \min(|\ \alpha_g - w_{pg}\ |, w_{pg}) \qquad \text{where } \alpha_g \leq 1 \tag{4}$$

and

$$\mu_g = \mu_0 \left(1 - \frac{2}{\pi}\text{atan}\left(10 * \left(\frac{R_c(s_i)}{R_d(s_i)}\right)^{\delta_r}\right)\right) \qquad \text{where } \mu_0 = 0.3 \tag{5}$$

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

$$w_{pg} = w_{pg} + \delta_p * \Delta_a. \tag{6}$$

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

$$V_i = \frac{P_i + (\Delta P * (t_{mot} + t_{dist}))}{(t_{mot} + t_{dist})^2} \tag{7}$$

where ΔP is the estimated change in pain over 1 cycle. 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 perform 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 (7) depends on the number of pains above threshold.

3 NeoAxis Implementation

A cognitive architecture organization based on the ML idea was introduced in [19]. This architecture uses a physical body in a physical environment. However, making a physical robot costs money and a design effort, so it is very helpful to use a computer simulation that can imitate real-time physical conditions. Thus we used a simulated environment with a virtual robot.

We implemented the infrastructure of the OML agent in NeoAxis describing the motivated agent functionality in C++ and C#. The virtual environment for OML agents built in NeoAxis is a 3D simulated world governed by realistic physics to

present the robots within 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 works in the created environment, discovers these rules and learns to use them to its advantage.

In this environment, we created resources that the agent could use, and endowed the agent with the ability to act on these resources (listed in Table 1). The agent's actions are driven by its pains. Only two pains, hunger, which increases over time, and curiosity, are predefined. Curiosity pain makes the agent explore the environment when no other pain is detected. Other pains are learned by the agent using the goal creation methodology [18]. The agent observes which resources it needs and introduces abstract pain(s) if they are unavailable. We also defined world rules, which describe which agent actions make sense and what their results are. Those rules are listed in Table 1 and are unknown to the agent. 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 (Primitive)
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

To 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 created a simple, but effective test bed within the NeoAxis engine.

The basic steps of OML agent simulation in NeoAxis contain successive iterations:

Each iteration consists of the Agent Phase, where the agent observes the Environment, updates its internal state, and generates motor output, and an Environment Phase where the environment performs the agent's action and updates itself accordingly, e.g. updates resource quantities, objects' locations, determines motor action outcome, etc.

To visualize resources quantities, current task, pains levels and the agent's memory, we added windows that 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 the pains display is red. In this screenshot (Fig. 2), the agent action is driven by the 'Lack of Money' pain. Sometime the agent takes a nonsense action like "Play for joy with hammer." Nevertheless, even actions such as this are used to learn. The memory window, presented in Fig. 2 on the left, displays the memory array state. When the color is gray, then this means that the agent has not learned usefulness of this action yet. When the color is white then this means that the action is useful. Each row in the memory array corresponds to a driving pain, while each column represents a possible action.

Any action performed by the agent is based on the “action tree” that tells the agent what elementary actions must be used to accomplish a desired composed action. For instance, if the agent decided to eat an apple, it must first walk in the direction of the apple, pick it up, and eat. The agent selects the paths to the desired object, while avoiding obstacles along the way. An OML agent is capable of estimating how long it will take for it to approach the object based on the distance information. It uses this information to select the most appropriate action based on the state of the environment and its own internal state. The agent is informed by its sensory inputs when he approached the desired object so it can start the desired action.

We have run multiple simulations where we modified resource quantities and motor action times. When starting resources were sparse, the agent couldn't learn all valid actions because it ran out of resources to test new actions. When resources, like food, were plentiful, the agent did not bother to learn anything new once the hunger pain was under control. Also, when action times were too long, the agent couldn't satisfy all pains. However, with parameters adjusted so that the agent doesn’t have access to too many or too few resources the OML agent learns correct behavior outperforming any reinforcement learning agent.

Figs. 3 provides results from the OML agent simulated in Matlab showing dynamic changes in various pain levels.

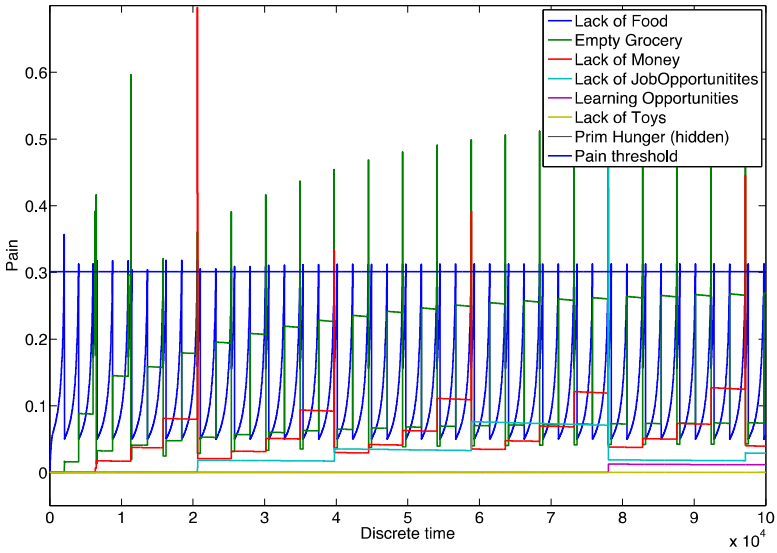

Fig. 3. Matlab simulation results showing dynamic changes in the pain levels

Only the primitive pain (hunger) is predefined and is gradually increasing. As the agent learns how to reduce the pain (by eating food) it introduces an abstract need to have food, and if food is in a short supply the pain of not having it increases. To supply food the agent learns more advanced skills (like buying food) and introduces abstract needs and related abstract pains. These abstract pains were unknown to the agent before it discovered how to use an abstract resource (e.g. money) to its advantage. We can see this in the simulation plot on Fig. 3 by observing that there was no pain of lack of money before 5000 iterations.

The simulation demonstrates that the agent was able to manage its various needs. Once abstract needs were introduced and related pains defined, the agent learned how to keep all pains below threshold, set in our simulation to 0.3.

Most of our code was initially prototyped in Matlab and then ported to NeoAxis, due to the ease to code and familiarity we have with developing Matlab applications. It should be noted, that the Matlab version is not strictly equivalent to the NeoAxis version, since the NeoAxis simulation operate in real time. It adjusts the decision making process to consider time delays required to approach various objects and the time needed to act on them. Thus, initial resource levels and motor times were adjusted to allow the NeoAxis simulation to operate efficiently. As a result of the changes made to the simulation parameters, the NeoAxis agent's pain passes threshold more quickly than in the Matlab simulation, effectively accelerating the rate at which it learns. However, despite the difference in the setup of the two simulations, the underlying model is identical, and as can be observed from the charts, the agent is able to effectively learn the desired actions.

Real time simulation showing the agent's actions and changes in the pains, observed in the display windows (Fig. 2), is in agreement with the Matlab results.

4 Conclusions

In this paper, we have presented our work on motivated learning agents with a focus on simulating the agent within a graphical environment. We included modifications to original ML algorithm [18], with 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 δ_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 virtual environments for our agents to interact with.

The simulation results of the embodied OML agent in the virtual 3D environment in NeoAxis prove that our theoretical assumptions for motivated learning agent memory organization, determination of bias signals, weights, goal creation and selection, as well as 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.

Acknowledgements. This work was supported by the grant from the National Science Centre DEC-2011/03/B/ST7/02518.

References

1. Hirukawa, H., et al.: Humanoid robotics platforms developed in HRP. Robotics and Autonomous Systems 48(4), 165–175 (2004)
2. Bakker, B., Schmidhuber, J.: Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization. In: Groen, F., Amato, N., Bonarini, A., Yoshida, E., Köse, B. (eds.) Proceedings of the 8th Conference on Intelligent Autonomous Systems, IAS-8, Amsterdam, The Netherlands, pp. 438–445 (2004)
3. Oudeyer, P.-Y., et al.: Intrinsically Motivated Exploration for Developmental and Active Sensorimotor Learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 107–146. Springer, Heidelberg (2010)
4. Ro, S., et al.: Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning. In: IEEE Intl. Conf. on Robotics and Biometrics, pp. 665–670 (2009)
5. Singh, S., Barto, A.G., Chentanez, N.: Intrinsically motivated learning of hierarchical collections of skills. In: Proc. 3rd Int. Conf. Development Learn., San Diego, CA, pp. 112–119 (2004)
6. Laird, J.E.: Extending the Soar cognitive architecture. In: Artificial General Intelligence 2008, pp. 224–235. IOS Press, Memphis (2008)
7. Anderson, J.R., et al.: An integrated theory of the mind. Psych. Review 111(4), 1036–1060 (2004)
8. Langley, P., Choi, D.: A unified cognitive architecture for physical robots. In: 21st Nat. Conf. Artificial Intelligence, pp. 1469–1474. AAAI Press, Boston (2006)
9. Baars, B.J., Franklin, S.: Consciousness is computational: the LIDA model of global workspace theory. Int. J. Machine Consciousness 1(1), 23–32 (2009)

10. Cassimatis, N., Nicholas, L.: Polyscheme: A Cognitive Architecture for Integrating Multiple Represetnation and Inference Schemes, MIT Ph.D. Diss. (2002)
11. Sun, R.: The importance of cognitive architectures: an analysis based on CLARION. J. Experimental and Theor. Artif. Intell. 19(2), 159–193 (2007)
12. Brooks, R.A.: Intelligence without reason. In: Proc. 12th Int. Conf. on Artificial Intelligence, Sydney, Australia, pp. 569–595 (1991)
13. Pfeifer, R., Bongard, J.C.: How the Body Shapes the Way We Think: A New View of Intelligence. The MIT Press, Bradford Books (2007)
14. Starzyk, J.A.: Motivation in Embodied Intelligence. In: Frontiers in Robotics, Automation and Control, pp. 83–110. I-Tech Education and Publishing, Austria (2008)
15. Clark, A.: Reasons, Robots and the Extended Mind. J. Mind and Language 16(2), 121–145 (2001)
16. Kanda, T., et al.: Development and evaluation of interactive humanoid robots. Proceedings of IEEE 92(11), 1839–1850 (2004)
17. Weng, J., et al.: Autonomous mental development by robots and animals. Science 291(5504), 599–600 (2001)
18. Starzyk, J.A.: Motivated Learning for Computational Intelligence. In: Igelnik, B. (ed.) Computational Modeling and Simulation of Intellect: Current State and Future Perspectives, ch.11, pp. 265–292. IGI Publishing (2011)
19. Starzyk, J.A.: Mental Saccades in Control of Cognitive Process. In: Int. Joint Conf. on Neural Networks, San Jose, CA, July 31-August 5, pp. 495–502 (2011)