A Generic Architecture for Adaptive Agents Based on Reinforcement Learning

Philippe Preux[a,*,1], Samuel Delepoulle[a], Jean-Claude Darcheville[b]

[a] *Laboratoire d'Informatique du Littoral (LIL), UPRES-JE 2335, Université du Littoral Côte d'Opale, B.P. 719, 62228 Calais Cedex, France*

[b] *Unité de Recherche sur l'Évolution des Comportements et des Apprentissages (URECA), UPRES-EA 1059, Université de Lille 3, B.P. 149, 59653 Villeneuve d'Ascq Cedex, France*

Abstract

In this paper, we present MAABAC, a generic model for building adaptive agents: they learn new behaviors by interacting with their environment. These agents adapt their behavior by way of reinforcement learning, namely temporal difference methods. MAABAC is presented in its generality and then, different instantiations of the generic model are presented and experiments are reported. These experiments show the strength of this way of learning.

Key words: modeling, dynamics of behavior, adaptive systems, reinforcement learning.

We want to model and simulate certain aspects of the dynamics of the behavior of living organisms. There is a growing interest of researchers working on living systems for virtual laboratories in which they can put their models to the test and perform virtual experiments while being able to control each parameter. Mostly considering population dynamics at its origins, virtual laboratories are now dealing with systems of increasing complexity such as whole ecosystems, societies, or molecular interactions. As involving collections of interacting entities, multi-agent systems (MAS) [1] generally lye at the heart of the implementation of virtual laboratories. There are works dealing with software simulations and others dealing with robots. Both approaches have pros and cons. For example, a robot action is inherently slow with regards to a computer simulation; a robot is brittle; and it takes some time to build. However, it is very difficult to simulate accurately the laws of physics in software and, in many circonstances, physics solve certain problems in real systems. Clearly, everything that happens in a software simulation has been programmed; this is not true for robots where physics can have a large effect. This also implies that we can control every parameter in a computer simulation which helps in tracing back the undergoing of a simulation and fully understand its outcome. Furthermore, a software experiment can be reproduced many times identically, while this is not true with robots in which noise and friction have random effects.

* Corresponding author.
Email addresses: `ppreux@univ-lille3.fr` (Philippe Preux),
`delepoulle@lil.univ-littoral.fr` (Samuel Delepoulle),
`darcheville@univ-lille3.fr` (Jean-Claude Darcheville).
[1] Current affiliation: Groupe de Recherche en Apprentissage Automatique (Machine Learning Research Group), Université de Lille 3, France

In our case, we are particularly interested in how the behavior of one animal evolves during its lifespan. This evolution is the product of two interacting processes: the development of the organism, and its interaction with its environment. Here, we focus on this second point, that is the study of the dynamics of behavior due to the interaction of the organism with its environment.

The model relies on a well-established law, observed over very numerous experiments in the fields of psychology and ethology, namely, the principle of the selection of behaviors by their consequences (SBC principle) [2]. Originally known as the law of effect and dating back to the end of the XIX^{e} century [3], this law states that the frequency of emission of a certain behavior increases if this behavior is followed by favorable consequences. Despite its simplicity, some researches argue that it is at the very root of all behaviors emitted by any living being, even those qualified as cognitive (see e. g. [4,5]). Our aim is to investigate this hypothesis using the tool of computer simulations.

From the psychologist standpoint, one practical goal is, using minimalist hypothesis, to give an account of various observed behaviors. To this end, computer simulations constitute a very powerful tool to explore the dynamics of a set of interacting agents. Computer simulations require a formalized model of what is to be simulated. In itself, this formalization is already worthwhile. The SBC principle is rather vague: it may lead to various formalizations. One decision to take at an early stage of the design of the model is its level of description. Indeed, the SBC principle has been described at the level of behaviors, but the notion of behavior is not so clear. We focus on the dynamics of behavior so that the level of description should let it change naturally along time. So, it can not be something so elaborate as “walk, grab an object, see, ...”. Having demonstrated the reality of the SBC principle at the neuronal level, Edelman and co-workers have studied this issue at the level of neurons [6]. At this point, it should be made clear that we consider the SBC principle as the learning mechanism at work in living organisms. In the case of vertebrates, considering that behaviors are the result of the activity of muscles acting on a skeleton, we have developped a model in which the key component is the muscle. This led us to a behavioral model which is described in section 1. The adaptation, or learning, mechanism of the muscle is Q-Learning which belong to the family of temporal difference (TD) algorithms that perform reinforcement learning. TD methods are outlined in section 2. Section 3 details some of the simulations that have been performed. Finally, section 4 is a discussion of this work, addressing the relevancy of the model, usefulness and limits of this research, and current and future lines of work.

1 The behavioral model

While lots of models of animal behavior rests on neuronal activity (see *e.g.* [6,7]), we adopt a behavioral model. Before going any further, let us precise what we mean by a "behavioral model", and "behavior". Regarding neuron based models, we have a rather clear idea of what a neuron is, at least in the case of formal neurons. Neuronal activity is thus clearly defined. The notion of behavior is more elusive and we define it with regards to our aim of studying its evolution. We can define various levels of behavior, ranging from low level (neuron activity) to high level (e.g. playing chess, or going to the work place) corresponding to highly cognitive processes. Being at the center of our concern, the SBC principle requires that a behavior can be slightly modified by some sort of random process. This led us to design a model at the level of muscle activity. In this paper, the term “muscle” refers to this “tissue composed of fibers capable of contracting to effect bodily movement” [2] in conjunction with its controlling motoneurons. We do not aim at using a particularly precise simulation of a muscle in a biological or physiological sense. We consider a muscle as an agent that can pull a bone around a joint. Basically, a muscle can get contracted more, get relaxed or remain in its current state of contraction. For example, in 2 dimensions, a certain articulation is controlled by two muscles, one that increases the angle of flexion of the articulation, while the other one decreases this angle; the actual angle of flexion is simply related to the difference of contraction of the two muscles. An animal is then considered as a collection of bones and muscles. It may have perceptive organs but we have not focussed very much on perception yet.

Our goal is to define a generic architecture to model organisms that continuously adapt their behavior from interacting with their environment. These organisms are made of segments and joints. Fig. 1 illustrates two possible forms. We call MAABAC this family of organisms. It has been originally introduced in S. Delepoulle’s PhD dissertation [8]. This section describes this generic architecture and exemplifies it with an instance of it.

1.1 The Generic MAABAC Architecture

Basically, the architecture of MAABAC is a non supervised MAS in which the behavior of each agent is controlled by a Q-Learning. In MAABAC, an agent represents a muscle as explained above. Each agent has a certain level

[2] following the definition found in *The American Heritage College Dictionary*, Houghton Mifflin, 1993

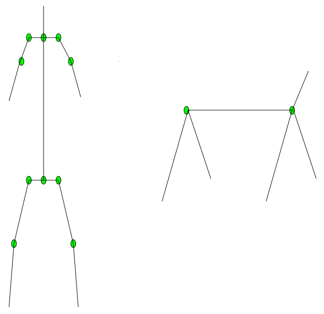

Figure 1. Visual aspects of some instances of MAABAC. We see bones (segments) and joints (dark discs).

of contraction $c \in [0, N[$ which defines its current state. Each muscle can exhibit three behaviors: maintain its current level of contraction, increase it by 1, decrease it by 1. At each time step, the contraction involves a certain energetic consumption modeled by a negative return of value $-\kappa(\frac{c}{N})^2$ with $\kappa \in]0, 1]$.

The overall architecture is sketched at Fig. 2. Basically, there is a set of muscles which are fed by the output of a "perception" agent itself fed by the "environment" agent. The action of the muscle agents implies a certain configuration of muscles which in turn implies a modification in MAABAC's position. The effect of the muscle configuration on MAABAC's position also depends on physical parameters of the environment. MAABAC's movements involve changes in the environment.

It is important to notice that a muscle has absolutely no information regarding the other muscles. It only knows its own current level of contraction and its current action. The perception agent also feeds the muscle agent with its perception of the environment. This perception may be void or very rich. For the moment, we have only worked with no perception or a very poor perception of MAABAC's position. The reason why we use a very poor perception is that

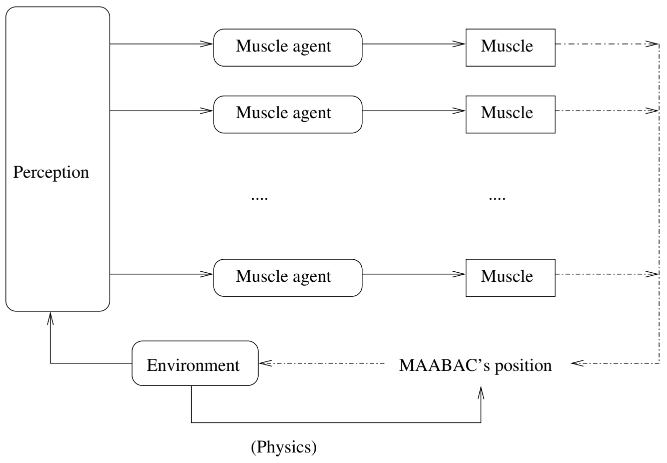

Figure 2. Outline of the generic architecture MAABAC. Plain line arrows indicate a transfer of information/data, while dashed line arrows mean “imply a modification”. Rounded rectangle are actual agents of MAABAC, while non rounded rectangle (muscles) are only figurative.

we do not want that MAABAC can perform its task by simply following some sort of gradient.

In the current implementation, agents are totally synchronized. That means that they perform their actions altogether, at a fixed pace. Afterwards, the software computes the new position of MAABAC to feeds back the muscles with the consequences of their action, again, all at the same time. The duration of a change is constant whatever its current state and previous actions having been performed. We do not think that such a synchronization may not have some effects on the dynamics of the simulation; in the same time, we are not aware of any work dealing with this aspects of things in real organisms. However, we remain aware of this because in some other works on simulation, we have experimented the crucial effect of synchronization of action in interacting agents [9].

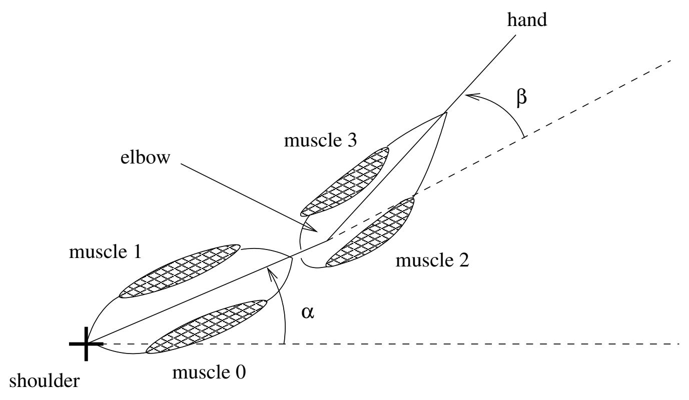

Figure 3. An arm in details.

1.2 An Instance of the Architecture

Now, we present an instance of the generic architecture. We consider an arm in 2D (see Fig. 3) made of two articulations (a shoulder which position is fixed) and an elbow. We name "hand" the free extremity of the arm. Each articulation is controlled by two muscles. The difference of contraction between antagonist muscles provides the angle of rotation of the articulation. The angles of rotation are given by:

$$\begin{cases} \alpha = \frac{\pi}{2}\frac{c_0 - c_1}{N} + \frac{\pi}{2} \\ \\ \beta = \frac{\pi}{2}\frac{c_2 - c_3}{N} + \frac{\pi}{2} - \alpha \end{cases}$$

where c_i is the level of contraction of muscle i. The amplitude of rotation is free: of course, it may be restrained to a certain range.

Following this description, we can generalize this architecture to any multi-segmented creature. Up to now, we have build a MAABAC made of one arm (MAABAC 1), a MAABAC made of two arms (MAABAC 2), and a MAABAC made of one body and 2 arms (MAABAC 3). The task that have been performed by these various versions is to put one hand, or two hands in a certain target zone. We are currently working on the addition of legs to let it learn to walk to reach with its hands a remote position.

Before detailing simulations, we describe the algorithm that controls agent behavior, namely Q-Learning.

2 Temporal Difference Methods

There is a family of algorithms that have been proposed as a model of the SBC principle, namely, temporal difference methods (TD methods) [10]. These belong to the realm of reinforcement learning algorithms. Proposed as early as 1959 in Samuel's checker player [11], they have recently been successfully applied to a certain number of problems where reactive learning algorithms are needed (scheduling in dynamical environments [12], exploration of the web [13], playing backgammon [14] to name a very few of them).

In reinforcement learning, an agent is situated in its environment on which it can act upon according to its perception of the state of the environment (which can be different from its "true" state). The actions of the agent leads to consequences on its environment, in particular, a new perception of its current state and a return. The return is central in this approach: it is generally implemented as a mere real number and the goal of the agent is to maximize its return along time.

So, let Σ be the set of states, $\Lambda(s)$ the set of all possible actions that can be emitted in state $s \in \Sigma$ (in full generality, both Σ and $\Lambda(s)$ can be variable along time). Assuming discrete time, at each time step $t \in \mathbb{N}$, being in a certain state $s(t) \in \Sigma$, a reinforcement learning algorithm learns which action $a(t) \in \Lambda(s(t))$ to emit in order to optimize the total amount of returns $R_T(t) = \sum_{k=0}^{k=T} \gamma^{t+k} r(t+k)$, where $r(t)$ is the return received at time step t, γ is a discount factor ($\gamma \in [0,1]$), and T can be finite (episodic task) or infinite. The discount factor aims at balancing the importance of short term consequences with regards to long term consequences: a small value gives a small importance to long term consequences, while a high value provides them much more importance.

This being set, a reinforcement learning algorithm learns a policy, that is which action to perform in any state in order to optimize its returns. There is a variety of algorithms to deal with this problem being more or less close to the resolution of Markov decision problems, thus dynamic programming [15]. Among them, TD methods are online algorithms that learn a policy by repeatedly interacting with their environment; then, if the environment is dynamic, the algorithm adapts its behavior. A TD algorithm learns expectation of the consequences of its actions in each state. After the emission of an action, it confronts its expectation with the actual return it got, and updates its expectation. These expectations are the average return it can obtain if in a certain state, it emits a certain action. This quantity is named the quality of the state/action pair: $Q(s, a) = E[R_T(t) \mid s(t) = s, a(t) = a]$ where $E[.]$ denotes the esperance of .. Then, a TD algorithm is basically made of a loop that, for the current time step t and the current state $s(t)$:

(1) selects an action $a(t)$ to perform,
(2) performs $a(t)$,
(3) observes the new state $s(t+1)$ and the return $r(t)$,
(4) updates its expectation $Q(s(t), a(t))$.

The selection of $a(t)$ is based on past experience; the idea is to select the action that is currently assigned the best quality in the current state. Naturally, a certain amount of hazard is included to let the algorithm explore new actions.

The update of qualities depends on the way qualities are represented in the algorithm. The most simple form is a look-up table which stores the current estimate of the quality for each state/action pair: this is the tabular version of the algorithm. However, when the state space is large, other structures should be used: a neural network, or more generally, a function approximator. These structures are much more compact than a table and they have greater abilities for generalization of learning. In our case, we use a tabular version because the number of states is rather small, namely tabular Q-Learning [16]. Using previously introduced notation, the update rule is:

$$Q(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \alpha[r(t) + \gamma \ \max_b Q(s(t+1), b) - Q(s(t), a(t))]$$

where α is the learning rate parameter. In a markovian environment, Q-Learning has been proved to converge asymptotically to the optimal policy under a few assumptions. In non markovian environments, no formal result is known regarding convergence. However, we have some experimental clues showing that Q-Learning and other TD algorithms can be used and perform well in non markovian environments.

Regarding the relevancy of TD to model the SBC principle, the adaptation of the behavior of a living organism is determined by the modification of the bio-electrical activity of certain neurons. In particular, evidences have been found of the role of dopamine neurons. Works in physiology have shown in the primate that the output of these neurons indicates the error in the prediction of reward, which is very close to what TD algorithms do [17,18].

MAABAC has no prior knowledge: the quality of its state/action pairs are all initially set to 0. At the beginning, each muscle is in a certain state of contraction. Once it has reached a certain goal (for example, putting its hand in a certain target zone), all the muscles receive a certain positive return; otherwise, the return to each muscle is only its (negative) consumption of energy. For a certain location of the target, there are generally many combinations of muscle contractions that put its hand in the target. The important point is that MAABAC performance depends on the combination of actions of all its muscles: there is no “best” action, nor any “best” state of contraction for any of them; there are only good combinations of their states of contraction, which result in a certain position of the arm. This means that each muscle should learn to emit correct behaviors without all the necessary information in hand. Furthermore, at a certain moment, if a muscle has been rewarded with a positive return (that is, the task has been solved) by emitting a certain action in a certain state, the next time it will be in the same state, emitting the same action is likely to provide no positive return (only a negative one due to energetic consumption) because the other muscles are no longer in the same state. This is thus a non markovian task.

3 Experiments

To assess the possibilities of MAABAC, the only way is to perform experiments. MAABAC is made of several coupled TD algorithms, each of which performing a non markovian task, and, in some of the experiments, the environment is not stationary. These properties are well beyond what we have yet been able to formalize analytically. So, we go on by describing experiments and the results obtained on various versions of MAABAC. We begin with the simplest version consisting of a single arm which shows surprisingly rich abilities. Then, we discuss the two-armed and the two-armed with a body versions of MAABAC.

Before detailing simulations, let us precise that each muscle is a tabular Q-learner which has $N = 50$ different states of contraction; thus, the Q-table is made of 150 real numbers. We use plain Q-Learning with Boltzmann selection of action, that is, the probability to select a given action a in the current state s is computed according to a Boltzmann probability distribution $e^{Q(s,a)/\tau}$, keeping a constant temperature $\tau = 0.5$. The constant κ is set to 1, γ is set to 0.9. The learning parameter α is kept constant at 0.5 while it is generally recommended to decrease it along learning. However, this recommendation holds for markovian tasks; in our case, it is not clear whether α should decrease or not and experiments have shown that our choice was correct. At the beginning of the simulation, the quality of each state/action pair is initialized to 0.

3.1 MAABAC 1: Experiments with One Arm

In this section, we present the experiments that have been performed with a MAABAC architecture reduced to a single arm, as presented above. Different experiments have been performed with it.

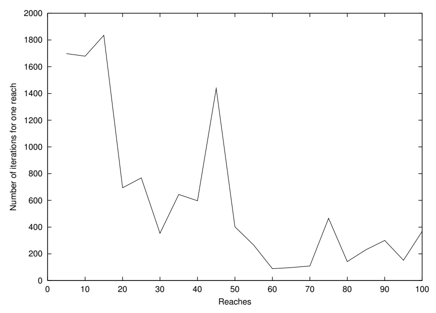

Figure 4. Learning to reach the target zone. The evolution along time (x-axis) of the number of iterations (y-axis) that are required to reach the target.

3.1.1 Learning to Reach the Target Zone

The first experiment that has been performed consists in checking whether this non supervised architecture is able to learn to control the arm and puts its hand in a target zone. This is a reaching experiment, a behavior that is rather well studied in psychology. Starting with no prior knowledge, we let MAABAC act and once it has put its hand in the target zone, we provide a positive return of value 1; otherwise, it does not get any positive return and each muscle receives its own negative return due to its energetic consumption. Once MAABAC has put its hand in the target zone, we reset its position and let it reach the zone again and, we iterate. We measure the number of iterations that have been performed for each reach. The word "iteration" refers here to one execution of the Q-learning loop as presented in section 2. The results are displayed at Fig. 4. It is clear that MAABAC learns to reach the target faster and faster: its initial attempts require approximately 1 700 iterations while after 50 reaches, it needs only a few hundreds. We also immediately see the non smooth and irregular shape of the learning curve which is typical in experiments on human beings in psychology.

Fig. 5 shows how MAABAC's behavior evolves along the reaches. At the very beginning, MAABAC's hand wanders in space. Formally speaking, this is a random walk resulting from the random activity of 4 random walkers. The randomness comes from the fact that qualities are initially set to 0. Along this wandering, the hand crosses the target zone but initially, it is not able to remain in it. However, MAABAC has learnt that there is some positive return to obtain somewhere. Once MAABAC has put its hand and let it remain a certain amount of iterations in the target zone, its hand position is reset to the initial position. Along subsequent reaches of the target, the movement becomes more and more straightforward towards the target until it is direct and smooth. Smoothness and directness are related to a minimization of the energy being used to perform the task: as the movement is being learnt, the consumption of energy decreases. Furthermore, there is an other reason for these two apparent properties. At the beginning of the simulations, the muscles are typically in a rather high state of contraction, the couple of muscles that controls the rotation of an articulation typically pulling strongly, involving a large energy consumption. Hence, at the beginning, MAABAC "does not know" that to obtain a rotation, it has to contract one of the two muscles, while relaxing the other one. Instead, it co-contracts both. Then, movements are difficult. After a few reaches, MAABAC has learnt to relax a muscle while the other is getting contracted; it has also learnt that, as long as the angle of flexion of an articulation is only due to the difference of contraction between the two muscles, the smaller the contraction, the smaller the energy consumption, for the same result. These features are very closely related to what is observed in babies and young children that learn new abilities such as walking, or writing with a pen: their muscles are very much contracted, leading to bad performances and energetic wastes. This is also observed with adults who learn new physical skills, for example in sport (skiing, ...). An other point that

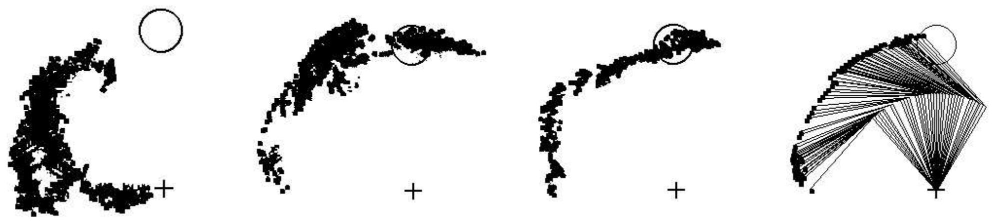
Figure 5. Different phases in the acquisition of the reaching movement by MAABAC 1 (time flows from left to right). In the 3 leftmost figures, each position of the hand is represented by a dot. The cross indicates the position of the shoulder and the circle indicates the location of the target zone. In the rightmost figure, all the positions of the arm are represented once a straightforward and smooth movement towards the target has been acquired (see text for more explanations).

is observed in MAABAC and in young babies is that the amplitude of rotation at the shoulder from the initial position to the target position is smaller that the amplitude of rotation of the elbow [19].

3.1.2 Extinction and Learning to Reach the Target Zone afterwards

After having learnt to reach the target zone, we first extinct this behavior by no longer providing the positive return when MAABAC puts its hand in the target zone. At the beginning, MAABAC keeps on putting its hand in the target zone. After a while, receiving no longer any positive return by so doing, its hand begins to wander in space. Once MAABAC is no longer aiming at putting its hand in the target zone, we provide the positive return again if it puts its hand in the target zone, without warning MAABAC in any way. Very quickly, MAABAC is able to take advantage of this renewed positive return and the movement does not need a lot of iterations to be re-emitted smoothly: this means that MAABAC had not forgotten the movement, it merely did not emit it because it was no longer worth its energetic cost. Fig. 6 displays the

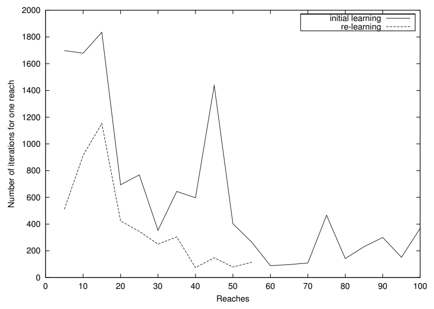

Figure 6. Comparison between MAABAC's initial learning curve (plain line: same as in fig. 5) and re-learning curve (dotted line) after extinction. It is clear that re-learning is much faster than the initial learning and this means that the behavior had not be forgotten after having been extincted.

number of iterations that are needed to re-learn the movement.

3.1.3 Shaping

Shaping is an extremely powerful method in animal training to make an animal emit a behavior that is naturally unlikely, such as making a bear ride a bicycle. The principle of shaping is the following: beginning with the animal natural repertoire of behaviors, by rewarding some of them, bring it little by little to the expected behavior. Using the idea of shaping in machine learning has already been proposed and studied a bit [20].

MAABAC's behavior can be shaped in various manners. Unlikely behaviors are those that cost MAABAC so much energy (provide such a large negative return) to obtain the positive return that they can not be learnt without any external intervention. Certain positions of MAABAC's arm suffer from this flaw: some of them require as much as 20 000 iterations to be reached. To make MAABAC learn this position of the hand, we propose to apply shaping: we let it learn to put its hand in a certain position as we have done earlier.

Then, we move the target zone a little bit towards the expected position and let MAABAC learn this new position. Once it has learnt this new position, a little bit more difficult than the previous one, we iterate the learning process. After 15 such different positions have been learnt following a gradient of difficulty, totalling 11 000 iterations, MAABAC is able to put its hand in the position which initially required 20 000 iterations. After this shaping, MAABAC is able to put its hand in the target zone in no more iterations than in an easily learnt position. Furthermore, MAABAC puts its hand in this position naturally, while it might have put its hand in this position initially only exceptionally and would have not done it again due to its cost.

Clearly, we have been able to shape MAABAC's behavior, and the usefulness of this technique is crystal clear. By carefully designing a strategy of shaping, we can teach any (possible) behavior to a MAABAC instance; it becomes then a matter of teaching organization.

3.1.4 Generalization

Generalization is of key importance in (machine) learning. Indeed, a system able to learn to solve a problem is of poorer interest than a system able to learn to solve a problem and which, in the same time, has learnt to solve similar/closely related tasks. It is then an important experiment to determine whether MAABAC is able to generalize its learning, and to which extent it can generalize what it has learnt. For this purpose, we devised the following experiment: after having learnt to reach a target zone, we move the target to more and more remote positions and measure the number of iterations MAABAC needs to reach the target zone.

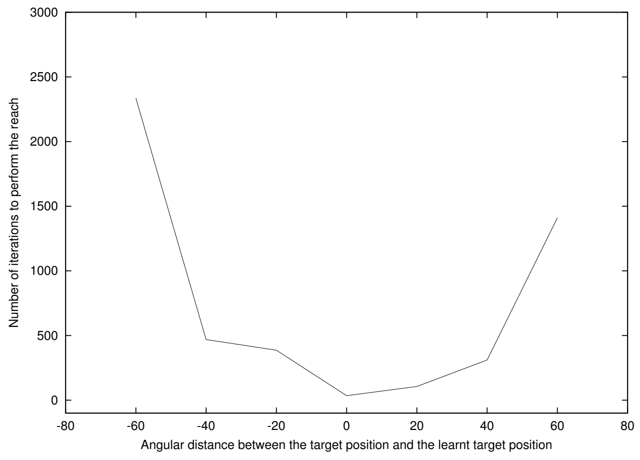

Figure 7. Generalization curve of the learning performed by MAABAC: we see that when the target zone is moved $\pm 40^o$ remote off the learnt position, MAABAC is able to generalize without any significant problem.

The result is displayed at Fig. 7. We see that MAABAC's performance remains very good when the target is moved within 40^o around the learnt position.

3.1.5 Other Tasks

Some other tasks have been performed. We report them here very briefly.

Once having learnt to reach a target zone, MAABAC is able to track a moving object that does not move too fast.

A certain zone in the space can be made very unpleasant: for example, one can imagine it figuring a much too hot zone. Putting the hand in this area leads to a very negative return. MAABAC seeks to escape this zone. Then, MAABAC learns to reach the target zone by avoiding this unpleasant zone. In the same order of ideas, if obstacle are present in the space, MAABAC learns to reach the target by moving the arm around the obstacle. In all these cases, MAABAC tends to reach the target zone by minimizing its energetic consumption.

Finally, if one muscle is made inactive, then the other muscles compensate so that the movement can still be learnt.

3.2 MAABAC 2: Two Arms

After experimenting one arm, we have experimented a MAABAC made of two arms (and named it MAABAC 2) [21]. Each arm has the same architecture as previously. There are thus 8 muscles to coordinate in MAABAC 2.

The task consists in putting either one hand, or both hands in a target zone.

Basically, all experiments that had been performed with MAABAC 1 have been performed with MAABAC 2. The results are equivalent. To speed-up learning, we have also experimented the following point: one arm having learnt a behavior in MAABAC 1, we use its Q-values to initialize the other arm Q-values. As expected, the results are good and, due to the fact that the algorithms are continuously adapting to their environment, the second arm can still finely tune its Q-values if it differs from the other arm (e.g., the length of segments are slightly different, muscle strength is different, ...).

3.3 MAABAC 3: Two Arms and a Body

We have then added a body to MAABAC which lets it orientate by rotating on itself [21]. One muscle controls the rotation on its axis, and we end-up with 9 muscles for the whole MAABAC made of a body and 2 arms (MAABAC 3).

Again, we have performed the same tasks which did not raise any problem. The interesting point we want to address here is that of the learning curve. As MAABAC is made more and more complex, one expects that the learning time to reach a zone or perform a certain task increases, very likely non linearly. However, the experimental results are in total opposition to this initial intuition: indeed, learning times decrease though MAABAC's complexity increases. When we compare the learning curve of MAABAC 2 with that of MAABAC 3 on the same task, MAABAC 3 requires approximately 2.5 times less iterations to learn the task than MAABAC 2. Speaking of computational time, since MAABAC 3 is made of 1.125 times more muscles, the net result is that MAABAC 3 learns the same task faster than the simpler MAABAC 2 does, though MAABAC 3 is more complex. This is an excellent point since we want to go on and make MAABAC more complex: we may have sub-linear scaling properties regarding learning times against the number of agents.

4 Discussion

In this section, we want to discuss what we have done and reported in this paper, as well as indicate our current research paths.

The definition of the MAABAC architecture aims at providing us with a generic tool to study behavior learning in living systems and in artificial systems. Based on living systems, our key hypothesis is that a system learns new behaviors because of its interactions with its environment which selects its behaviors. This process leads to a tuning of behaviors, as well as so-called "new" behaviors. A living organism emits a new behavior in response to environmental "solicitations", that is, to fit better in its environment. The environment can involve physical constraints, as well as social constraints if the organism is interacting with other adaptive organisms.

From the experiments that we have done yet on more and more complex instantiations of MAABAC, we draw the conclusion that any behavior can be learnt as far as it is physically possible for MAABAC to emit it, and that this behavior is favored by the environment. The very powerful shaping technique can be used to acquire unlikely behaviors by progressively bringing the artefact from its initial behavioral repertoire to the expected behavior. In complement with shaping, we have experimented other key features of animal behavior such as extinction of behavior, escaping, avoidance, and generalization. We have compared MAABAC's behavior with the behavior of young babies as described by [22,23]. Different features have been found similar in how MAABAC acquires the behavior and how babies do. This yields interesting feedback towards psychology.

Regarding existing works, one key originality of our work is that we do not want to reproduce a behavior by any means but, 1) we want to reproduce the acquisition of behaviors and the adaptation of behavioral repertoires during the lifespan of living organisms, and 2) we want to use hypothesis that are relevant to animal and human learning works in psychology and ethology. To date, most works have dealt with performing a certain behavior such as walking (see e.g. [24]), or evolving a control architecture to perform a certain behavior (see e.g. [25]). Some other works put the emphasis on learning, either by incrementally building a learning artefact [26], or by buidling an adaptive architecture that, for instance, can learn by imitation [27]. The grounding on hypothesis relevant to living organisms is more original. [28] studies the reaching with Q-learning but using a different approach as ours. Despite modelling the arm with a collection of muscles, reaching is considered as a markov decision problem and Q-learning is used to solve it. A noticeable work is that of G. Edelman on the Darwin robot family [6]. In particular, the Darwin III is made of an eye and an arm aiming at learning to reach a target zone. While we perform the same kind of learning tasks by a functional modeling of behavior adaptation, Edelman's team uses a neurobiological based model which is much larger than ours. Furthermore, Darwin III has a vision which provides a digitalized image of its environment, thus the target. Moreover, the computational requirements are totally different, much heavier in the case of Darwin than for our MAABACs. Finally, from a conceptual point of view, Darwin III relies on the idea of competition between agents (groups of neurons) while MAABAC relies on the cooperation of agents.

At this stage of research, we, as other teams, have to face the problem that when new behaviors are learnt, there is a tendency that after a certain amount of learnings have been done, earliest learnings tend to fade away if they have not been activated for a long period of time. In MAABAC, learning one behavior leads each agent to solve a non markovian problem which is not an obvious problem. As others, we think that to solve the many task learning problem without loss of non recently refreshed ones, the creation of new states as well as the modification of existing states in the learning algorithm might be a solution. However, these notions which are related to dynamically reconfigurable architectures are not mastered today: it is a field of active research and we are working on this idea.

One might be doubtful about our approach of learning by interacting with the environment and whether MAABAC will ever be able to learn anything else than merely responding to stimuli. The idea that animal learning is based on the interaction of the animal with its environment is well argued and very complex behaviors, including social behaviors, can be explained in this way. Furthermore, it is also well known in this field that the richer and the more complex the environment that is perceived by an organism, the richer and complex its behaviors and behavioral repertoire. So, we think that making MAABAC's environment more complex and making MAABAC able to perceive it will lead to complex behaviors.

We are also considering the issue of teaching MAABAC new behaviors in order to accelerate its learning. This mixture of reinforcement learning and supervised learning leads to different interesting technical and fundamental problems. One of the major flaw of reinforcement learning is that it is slow at the beginning when it starts no prior knowledge. Such guiding is a relevant approach to alleviate this problem. We are currently working on this issue in the context of the generation of sequences of animated images. Furthermore, this hybrid model of learning may provide a model of animal learning under teaching.

One track of research might be to let interact two or more MAABACs, eventually having different forms (see e.g. the two forms in Fig. 1), and see whether social behaviors emerge from this interaction. Existing experiments in robotics gives us the intuition that interesting dynamics will indeed emerge from this.

Regarding the model, one issue concerns its architecture and whether a multi-agent architecture is better than a single agent controlling all muscles. From the point of view of the application, the choice we made is natural: one muscle equals one agent. Then, as in a developping living organism, muscles have to organize their activities to produce global behaviors that are relevant for the animal. So, the use of a multi-agent system involves the need for a self-organization of the agents. To put the system in a worst case, the agents of MAABAC have no information with regards to each other. It is likely that correlations develops in the brain of the animal while it acquires its behaviors. However, very little is known precisely on this point and consequently, no such correlation is used in our simulation.

From the restricted point of view of pure computing efficiency, the issue regarding how learning is achieved in a society of agents that are not sharing any information in comparison with agents which exchange some information is currently being addressed in different works. Stemming from game theory [29,30,31], one distinguishes independent learners (IL) which determine their action on their own, from joint action learners (JAL) which determine their action on the basis of other agent policies [32]. In MAABAC, the muscles are ILs. No definite answer is available regarding which solution is the best.

From the point of view of relevancy with regards to real organisms, if correlations between the action of muscles is somewhat innate to a certain extent, they also get acquired when the behavior is emitted repeatedly [33,34]. One argument for this point is the decreasing amount of co-contractions of opposite muscles when one learns a new behavior. Regarding the innate part, we might accomodate it by a careful initialization of the Q tables which would avoid initial random wanderings of the hand. However, details regarding these innate abilities are lacking.

At this point, it is time to discuss reinforcement learning and temporal difference methods as a relevant model of animal adaptation of behavior. Designed to this end, there are physiological arguments in its favor [17,18]. After this work, we feel pretty confident that we could go on and complexify MAABAC and obtain, eventually after large running times of simulation, the acquisition of complex bahaviors. To this end, shaping as well as guiding could be use to decrease the learning times. We do not feel that the real problem is here. To our eyes, the real problem is at least two-fold. First, there is the issue of simulating the physics of the environment accuretaly enough; animal develops and adapts its behavior in a physically grounded environment, and physical laws have effects on this. However and very unfortunately, the modeling of the effect of a physical enviromnent on an artificial creature is not mastered yet for tasks related to locomotion, such as walking, swimming, ... Second, if temporal difference methods provide relevant dynamics, they obviously strongly rely on the definition of states that is used; what is a state? In this work, we consider a state as a level of contraction of a muscle. This is rather crude. In particular, some dynamical effects such as the acceleration and deceleration of movements is not modeled whereas it is kown that infant's movements are constantly accelerating or decelerating during reaching [28]. This dynamics can have, at least, two sources: either physics by way of inertia, or the fact that the speed of movement is encoded in the state. In the future, light will have to be shed on these issues if we want to be able to generate automatically by computer sequences of images and make sequences of movies in which behaviors are correctly modeled.

Finally, we think that this work which initially concerns modeling and simulation may, and will, have various engineering applications. In particular, we have currently begun working with researchers in bio-mechanics in order to generate sequences of images automatically. The interest of using algorithms mimicing nicely the dynamics of behavior of real organisms is expected to lead to “natural” movements of organisms.

Acknowledgements

During the realization of this work, Samuel Delepoulle has been partially funded by contract 97 53 0283 from “Conseil Régional Nord-Pas de Calais, France”. This work has also been funded by the french “Programme Cognitique”. Authors also acknowledge the comments of three anonymous referees.

References

[1] J. Ferber, Multi-agent systems, Addison-Wesley, 1999.

[2] B. Skinner, Selection by consequences, Science 213 (1981) 501–514.

[3] E. Thorndike, Animal intelligence: An experimental study of the associative process in animals, Psychology Monographs 2.

[4] F. Toates, What is cognitive and what is not cognitive, in: [35], 1994, pp. 102–107.

[5] E. Spier, D. McFarland, Learning to do without cognition, in: [36], 1998, pp. 38–47.

[6] G. Reeke, O. Sporns, G. Edelman, Synthetic neural modelling: Comparisons of population and connectionist approaches, in: R. Pfeifer, Z. Schreter, F. Fogelman-Soulié, L. Steels (Eds.), Connectionism in Perspective, Elsevier Science Publishers, 1989.

[7] H. Frezza-Buet, F. Alexandre, From a biological to a computational model for the autonomous behavior of ana animat, Information Sciences 144 (2002) 1–43.

[8] S. Delepoulle, Coopération entre agents adaptatifs ; étude de la sélection des comportements sociaux, expérimentations et simulations, Ph.D. thesis, Université de Lille 3, URECA, Villeneuve d'Ascq, thèse de doctorat de Psychologie (Oct. 2000).

[9] S. Delepoulle, P. Preux, J.-C. Darcheville, Dynamique de l'interaction, in: B. Chaib-Dra, P. Enjalbert (Eds.), Proc. Modèles Formels de l'Interaction, Toulouse, 2001, pp. 141–150.

[10] R. Sutton, A. Barto, Reinforcement learning: an introduction, MIT Press, 1998.

[11] A. Samuel, Some studies in machine learning using the game of checkers, IBM Journal of Research and Development 3 (1959) 211–229, reprinted in E.A. Feigenbaum and J. Feldman (eds), *Computers and Thought*, pp. 71–105, Mc Graw-Hill, New York, 1963.

[12] W. Zhang, T. Dietterich, A reinforcement learning approach to job-shop scheduling.

[13] J. Rennie, A. McCallum, Using reinforcement learning to spider the web efficiently, in: Proc. ECML, 1999.

[14] G. Tesauro, Temporal difference learning and TD-Gammon, Communications of the ACM 38 (1995) 58–68.

[15] D. Bertsekas, J. Tsitsiklis, Neuro-dynamic programming, Athena Scientific, 1996.

[16] C. Watkins, Learning from delayed rewards, Ph.D. thesis, King's college, Cambridge, UK (1989).

[17] W. Schultz, P. Dayan, P. R. Montague, A neural substrate of prediction and reward, Science 275 (1997) 1593–1599.

[18] R. Suri, W. Schultz, Temporal difference model reproduces anticipatory neural activity, Neural Computation 13 (4) (2001) 487–494.

[19] E. Thelen, L. Smith, A dynamic systems approach to the developpement of cognition and action, MIT Press, 1994.

[20] M. Dorigo, M. Colombetti, Robot Shaping: An experiment in behavior engineering, MIT Press, 1998.

[21] C. Cassagnabère, Modeling and simulation of the adaptive behavior of a virtual arm during a reaching movement, Master's thesis, Université du Littoral Côte d'Opale, Computer Science Dpt, France (2001).

[22] E. Thelen, L. Smith, A dynamic systems approach to the developpement of cognition and action, MIT Press, 1994.

[23] C. Boyer, J.-C. Darcheville, Interaction organisation-environnement dans l'émergence de la saisie chez le nourissons, in: Congrès National de la Société Française de Psychologie, 1999.

[24] R. Brooks, Cambrian Intelligence: The early history of the new AI, MIT Press, 1999.

[25] J. Kodjobachian, J. Meyer, Evolution and development of neural controllers for locomotion, gradient-following, and obstacle avoidance in artificial insects, IEEE Transactions in Neural Networks 9 (1998) 796–812.

[26] G. Metta, R. Manzotti, F. Panerai, G. Sandini, Development: is it the right way towards humanoid robotics?, in: IAS-6, 2000.

[27] P. Andry, S. Moga, P. Gaussier, A. Revel, J. Nadel, Imitaiton: Learning and communication, in: J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblatt, S. Wilson (Eds.), Proc. of From Animals to Animat Conf.: Simulated Adaptive Behavior, MIT Press, 2000, pp. 353–362.

[28] N. Berthier, Analysis of reaching for stationary and moving objects in the human infant, in: Neural networks models of complex behavior – Behavioral Foundations, Elsevier, 1997, pp. 283–301.

[29] M. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Proceedings of the 11th International Conference on Machine Learning (ICML-94), Morgan Kaufmann, New Brunswick, NJ, 1994, pp. 157–163.

[30] T. Sandholm, R. Crites, On multiagent Q-learning in a semi-competitive domain, in: Workshop on Adaptation and Learning in Multiagent Systems at the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, 1995, pp. 71–77.

[31] J. Hu, M. Wellman, Multiagent reinforcement learning in stochastic games, (Submitted for publication.) (1999).

[32] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative learning, in: M. N. Huhns, M. P. Singh (Eds.), Readings in Agents, Morgan Kaufmann, San Francisco, CA, USA, 1997, pp. 487–494.

[33] C. von Hofsten, Eye-hand coordination in newborns, Developmental Psychology 18 (1982) 450–461.

[34] K. Ennouri, K. Dubon, C. Notides, H. Bloch, Visual control upon hand reaching movements in neonates, Infant Behavior and Development 17.

[35] D. Cliff, P. Husbands, J.-A. Meyer, S. W. Wilson (Eds.), From Animals to Animats 3 (Proc. of the third Int'l Conf. on Simulation of Adaptive Behavior), MIT Press, 1994.

[36] R. Pfeifer, B. Blumberg, J.-A. Meyer, S. Wilson (Eds.), Proc. Fifth International Conference on Simulation of Adaptive Behavior (SAB 5), MIT Press, 1999.