Exploration Tuned Reinforcement Function

Juan Miguel Santos [a] and Claude Touzet [b,1]

[a] *Departamento de Computación, F.C.E.yN., Universidad de Buenos Aires;*
Ciudad Universitaria, Pabellón I; 1428 Bs. As., Argentina.
Email:jmsantos@dc.uba.ar

[b] *Center for Engineering Science Advanced Research,*
Computer Science and Mathematics Division, Oak Ridge National Laboratory,
Oak Ridge, TN 37831-6355, USA. Email:touzetc@mars.epm.ornl.gov

---

Abstract

The aim of this work is to present a method that helps tune the reinforcement function parameters in a reinforcement learning approach. Since the proposal of neural-based implementations for the reinforcement learning paradigm (which reduced learning time and memory requirements to realistic values) reinforcement functions have become the critical components. Using a particular definition for reinforcement functions (RF), we solve, for a specific case, the so-called exploration versus exploitation dilemma through the careful computation of the RF parameter values. The proposed algorithm computes, during the exploration part of the learning phase, an estimate for the parameter values. Experiments with the mobile robot Nomad 200 validate our proposals.

Key words: Reinforcement function, reinforcement learning, robot learning, autonomous robot, behavior-based approach.

---

1 Introduction

Reinforcement learning (RL) dates back to the early days of cybernetics and work in statistics, psychology, neuroscience and computer science. In the last five to ten years, it has attracted rapidly increasing interest in the machine learning and artificial intelligence communities. Its promise is beguiling—a

---

[1] This research was funded in part by the Office of Energy Research, Basic Energy Sciences of the U.S. Department of Energy, under contract No.DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation, and in part during a previous appointement at DIAM-IUSPIM, Univ. of Aix-Marseille III, France.

Preprint submitted to Elsevier Preprint *1 December 1998*

way of programming robots by reward and punishment without the need to specify how the task (i.e., behavior) is to be achieved [3]. Reinforcement learning synthesizes a mapping function between situations and actions by maximizing a performance measure of the desired behavior. The reinforcement signal provided by the reinforcement function (RF) evaluates the entered situation relative to the desired behavior. The robot receives either positive or negative reinforcements according to the utility (i.e., desirability) of the situation entered as a consequence of the performed action. Reinforcement learning allows one, at least in principle, to bypass the problems of building an explicit model of the behavior to be synthesized or a meaningful learning base needed for supervised learning. There has been a lot of research related to the issues of convergence [2]; generalization [5]; exploration [8], [12]; and memorization [4]. A major milestone in the course of RL development has been the proposal of neural-based implementations, which reduced learning time and memory requirements to realistic values, allowing true RL applications [7]. Therefore, the RF has become the critical component and authors usually report the building of the RF as an emergent process involving lots of trials and errors. Despite all the ongoing research, there have been few efforts to propose a methodology, or define a robust set of heuristics, for the design of RFs. Ackley and Littman [1] introduce an Evolutionary Reinforcement Learning scheme where RL permits the individuals to improve their performance along their life. Thus, the performance measure of the complete system is indirectly used for accepting, or not, changes of the RF definition of each individual. Also, Mataric [6] proposes a careful RF design, based on the inclusion of progress estimators or/and sub-RF definitions. This way pure-delayed-reward (or very-delayed-reward) problems become delayed-rewards or immediate-rewards problems resulting in performance improvements.

In section 2, we introduce a particular definition of a RF. In section 3, we study the effect of the monotony property on the RF. In section 4, we propose an algorithm to compute, during the learning, the parameter values. In section 5, simulations and experiments involving the synthesis of avoidance obstacles are conducted with a mobile robot (Nomad 200). Finally, we discuss the obtained results and offer some conclusion.

2 A Definition of the RF

A RF should always imply positive, negative, and null reward [11]. If there is no positive reward, the evaluation function built during the learning phase will have “0” as maximum value and the policy cannot select effective actions. If there is no negative reward, the robot may remain in a dead-end situation forever. If there is no null reward, the evaluation function will be non-continuous at the frontier between positive and negative situation-action pairs; this will

Fig. 1. Each point of the graphs was obtained by performing 500 successive random moves for a mobile robot in an environment like the one displayed in fig. 4. a) Sum of positive rewards over the $\#ofit.$ versus the value of θ_+ [0 .. 1000], b) Sum of negative rewards over the $\#ofit.$ versus the value of θ_- [0 .. 2000].

affect the generalization quality. Thus, a desired ratio of positive and negative rewards over time during the exploratory part of the learning phase is mandatory. Let us consider the following RF expression, which has only one parameter per case (positive and negative rewards). $(s_1, ..., s_u)$ is the output readings of the sensors, $g_1\left(\right)$ and $g_2\left(\right)$ are any kind of functions.

$$RF(s_1, ..., s_u) = \begin{cases} +1 \ \ if \ g_1(s_1, ..., s_u) > \theta_+ \\ -1 \ \ if \ g_2(s_1, ..., s_u) < \theta_- \\ 0 \ \ otherwise \end{cases} \quad (1)$$

For the need of illustration, we take the case of an obstacle avoidance behavior synthesis for a mobile robot. How should we define the value of θ_+ (res. θ_-) so that it respects the desired ratios? As shown on figure 1, the sum of the rewards over the number of iterations ($\#ofit.$) value varies between 0.0 and 0.7 (depending on the values of the threshold θ_+). If θ_+ is small, the robot receives lots of positive rewards, and then the sum of the rewards will be large. It must not coincide with the total number of moves; otherwise, the robot would never receive negative rewards. The extreme case is when θ_+ is 0. Also, as we increase θ_+, the total amount of positive rewards diminishes. θ_- shows the opposite behavior respectively to negative rewards.

3 Reinforcement-Parameters Relation

In respect to the definition of the RF (eq. 1), we observe that a relation between the sum of received reinforcements and the parameter values exists. In particular, the sum of positive reinforcements ($\#r_+$) depends on the value θ_+ and the sum of negative reinforcements ($\#r_-$) depends on θ_-. We called this type of relation Reinforcements-Parameters Relation (RPR). Reinforcement and parameters are intimately related by the constraints present in the RF. These constraints, basically, relate a function $g_i()$ with a parameter θ_i by means of an order relation (e.g. the constraint $g_1(s_1, ..., s_u) > \theta_+$ in eq. 1).

In most real RL applications (e.g. behavior learning with real robots), the sensors may only partially perceive the environment. In this way, the constraints will be applied on estimations instead of the true value of the state variables. It is straightforward to show that the RPR associated with this form of the RF definition are monotonic, independent of the estimator characteristics and of the situation distribution during the learning.

4 Dynamic Estimation of Parameters in RF

Based on the monotony property of the RPRs, we have derived an algorithm to dynamically compute during the exploration part of the learning phase, the parameter values. The principle of the Update Parameters Algorithm (UPA) is described in figure 2.

Let us call the variables p_+ and p_- the desired values or ideal ratios (for example 0.2 for both values) that the real observations of the reinforcements $\#r_+$ and $\#r_-$ (over time) should match. The problem is to adjust dynamically the threshold parameters (θ_+ and θ_-) so that we observe the convergence of the real observations towards the desired ratios. In order to estimate the value of the real observations $\#r_+$ and $\#r_-$ over time, two variables $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ are defined. They are the number of positive rewards occurring in the last Δt iterations.

Δt is called the integration window (Step 1 of UPA). If the value $\frac{\#r_+^{\Delta t}}{\Delta t}$ is greater than p_+, then we have to modify θ_+ to obtain in the subsequent iterations a value of $\frac{\#r_+^{\Delta t}}{\Delta t}$ closer to the desired value p_+. If the relation between $\#r_+$ and θ_+ is decreasing monotonically, the value of $\Delta\theta_+$ is added to θ_+. In the opposite case, $\frac{\#r_+^{\Delta t}}{\Delta t}$ is smaller than p_+ and then, the value $\Delta\theta_+$ is subtracted from θ_+. If, instead, the relation is increasing monotonically the sign of $\Delta\theta_+$ must be inverted (first part of step 3 of UPA).

In the same way, if the value $\frac{\#r_-^{\Delta t}}{\Delta t}$ is greater than p_-, then we have to modify θ_- to obtain in the subsequent iterations a value of $\frac{\#r_-^{\Delta t}}{\Delta t}$ closer to the desired value p_-. If the relation between $\#r_-$ and θ_- is decreasing monotonously, the value of $\Delta\theta_-$ is added to θ_-. In the opposite case, $\frac{\#r_-^{\Delta t}}{\Delta t}$ is smaller than p_-

and then, the value $\Delta\theta_-$ is subtracted from θ_-. If, instead, the relation is increasing monotonously the sign of θ_- must be inverted (second part of step 3 of UPA).

If θ_+ (resp. θ_-) crossed the ideal value of θ_+ (resp. θ_-) the result of the subtraction between the estimation and p_+ (resp. p_-) will change sign. To detect these changes, in each iteration, the values of $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ are stored into Δr_+ and Δr_- respectively (Step 4 of UPA). Each time this crossing occurs, the absolute value of $\Delta\theta_+$ ($\Delta\theta_-$ resp.) is diminished by a factor k_1 (resp. k_2) (Step 2 of UPA).

The stop conditions are given by:

i) $\left|\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right| < \varepsilon$ and $\left|\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right| < \varepsilon$ during k iterations, $(0 < \varepsilon < 1)$,

or

ii) if the exploration phase has terminated. (Step 5 of UPA).

```
Initialization:
Define $\varepsilon$, $\Delta t$, $k$, $k_1$ and $k_2$ $(0 < k_1, k_2 < 1)$.
Define the values of $p_+$ and $p_-$.
Choose some initial value for $\theta_-$ and $\theta_-$.
Set the initial step for $\Delta\theta_+$ and $\Delta\theta_-$.
1. Compute $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ as
     $\#r_+^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, 1)$; $\#r_-^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, -1)$,
   where $\delta(x,y) = \begin{cases} 1 \ if \ \ x = y \\ 0 \ otherwise \end{cases}$.
2. If $(\Delta r_+ - p_+)\left(\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right) < 0$ then $\Delta\theta_+ = k_1\Delta\theta_+$ .
   If $(\Delta r_- - p_-)\left(\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right) < 0$ then $\Delta\theta_- = k_2\Delta\theta_-$ .
3. Set $\theta_+ = \theta_+ - sign\left(\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right)\Delta\theta_+$.
   Set $\theta_- = \theta_- + sign\left(\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right)\Delta\theta_-$ .

4. Store in $\Delta r_+$ and $\Delta r_+$ the current values of observer estimators,
     $\Delta r_+ = \frac{\#r_+^{\Delta t}}{\Delta t}$; $\Delta r_- = \frac{\#r_-^{\Delta t}}{\Delta t}$.
5. Check if $\left|\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right| < \varepsilon$ and $\left|\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right| < \varepsilon$ during $k$ iterations,
   where $k$ ¡¡ total number of iterations.
   If the condition is not satify, then go to 1.
```

Fig. 2. Update Parameters Algorithm. The expressions in step 3 depend on eq. 1.

Fig. 3. The Nomad 200 mobile robot.

5 Experiments

We have used the Nomad 200 robot (figure 3) in an experiment of synthesizing an obstacle avoidance behavior using its infra-red (IR) sensors. The typical arena is shown on figure 4. The Nomad 200 robot is represented inside the arena. The robot has 16 sensors uniformly distributed around its body.

The value returned by an IR sensor is 255 when there is nothing in front and 0 with a nearby obstacle (however, each value is coded on only 4 bits). The complete definition of the RF function is:

+1 if it is avoiding,

-1 if a collision occurs,

or 0 otherwise.

The robot is avoiding when the current sum of sensor values is greater than the previous one, the difference being greater than θ_+ and a collision occurs

Fig. 4. One of the arenas used in our experiments.

when the sum of the 8 front sensors is less than θ_-.This RF can be rewritten as:

$$RF((s_1,...,s_{16})^t,(s_1,...,s_{16})^{t-1}) = \begin{cases} +1 \; if \; g_1((s_1,..,s_{16})^t,(s_1,...,s_{16})^{t-1}) > \theta_+ \\ -1 \; if \; g_2(s_1,..,s_{16}) < \theta_- \\ 0 \quad otherwise \end{cases},$$

where $g_2(s_1,..,s_{16})^t = \sum_{i=1}^{4} s_i^t + \sum_{i=13}^{16} s_i^t$,

$$g_1((s_1,..,s_{16})^t,(s_1,..,s_{16})^{t-1}) =$$

$$= \left(g_2(s_1,..,s_{16})^t + \sum_{i=7}^{9} s_i^t\right) - \left(g_2(s_1,..,s_{16})^{t-1} + \sum_{i=7}^{9} s_i^{t-1}\right)$$

The sensors (5,6,10,11,12) are ignored because the initial form of the RF was defined for obstacle avoidance behavior synthesis with the miniature mobile robot Khepera [10]. Khepera's sensors are non-uniformly distributed and, since our goal is to automatically compute the threshold values —and not the expression of $g_i()$— we elect to reproduce the Khepera sensor distribution on the Nomad 200. Figure 5 shows the learning curve for θ_+ and θ_- during the exploration part with UPA. The initial values for θ_+ and θ_- have been set to 0.0 and 1000.0, $\varepsilon = 0$, $k_1 = k_2 = 0.99$, $\Delta t = 20$, $\Delta\theta_+ = \Delta\theta_- = 1.0$. The desired ratio of the observers $\#r_+$ and $\#r_-$ over time is 0.2 (for both). At the end of the UPA phase, the values for θ_+ and θ_- are 390.0 and 1150.0 (respectively). Figure 6 shows the behavior of $\#r_+/it.$ and $\#r_-/it.$ during the same experiment.

The oscillations correspond to changes in the sign of $\Delta\theta_+$ (or $\Delta\theta_-$) when the observers ($\#r_+$ or $\#r_-$ over time) crossed the desired value (p_+ or p_-). The

Fig. 5. θ_+ and θ_- values during the 500 iterations of the UPA phase. The initial values are set randomly to 0.0 and 1000.0, respectively.

Fig. 6. Values of $\#r_+/\Delta t$ and $\#r_-/\Delta t$ during the 500 UPA iterations. The oscillations are due to the limited size of the integration window (here $\Delta t = 20$ time steps) and the non-uniform distribution of the encountered situations. The objective is a ratio of 0.2 for the observers $\#r_+$ and $\#r_-$ over time.

magnitude of the oscillations in the curves of $\#r_+/\Delta t$ and $\#r_-/\Delta t$ vs. # of it. are due to the large size of the integration window which delays the parameter update. In order to attenuate this effect, we could diminish the size of the integration window (Δt). However, this value must not be too small, otherwise the observers will be greatly affected by the distribution of the encountered situations. In spite of the oscillations, if the values of $\Delta\theta_+$ (and $\Delta\theta_-$) are small enough and if the robot can carry out enough moves, then both the values of the observers and of the parameters converge.

After the 500 UPA iterations, Q-learning starts the pure synthesis of the behavior using a self-organizing map implementation [10]. During the learning phase, the neurons of the self-organizing map approximate the probability density function of the inputs. The inputs are the triplet: situation, the action and the associated Q-value (figure 7). The learning phase associates to each

Fig. 7. The self-organizing map implementation of the Q-learning.

Fig. 8. Selection of the best action to perform in the world situation. The Kohonen map is used as an associative memory: information is probed with part of it. a) The world situation and a Q-value of +1 are given as inputs. b) The answer is a winner neuron whose weights encode situation, action and the associated Q-value.

neuron of the map a situation-action pair plus its Q-value. It is a method of state grouping involving syntactic similarity and locality. The number of neurons equals the number of stored associations. The neighborhood property of the Kohonen map allows generalizing across similar situation-action pairs.

The self-organizing map is used in the following way: the best action to undertake in a situation is given by the neuron that has the minimal distance to the input situation and to a Q-value of value +1 (figure 8a). That is,

$d(i) = |situation - w_{situation,i}| + |1 - w_{Q-value,i}|$, where $-w_{situation,i}$ is the weight of neuron i associated to the situation input and $w_{Q-value,i}$ is the weight of neuron i associated to the Q-value input.

The selected neuron s corresponds to a triplet $(w_{situation,s}, w_{action,s}, w_{Q-value,s})$. It is this particular action= $w_{action,s}$ that should offer the best reward in the world situation (fig. 8b).

The effective action to be conducted by the robot will be the proposed action

Fig. 9. Graphs of $\#r_+/it.$ and $\#r_-/it.$ during the 200 learning + 100 test iterations.

plus a random component (to allow exploration). Then, the neuron corresponding to the situation and the action effectively performed is selected. The distance used is different from the exploration process. It includes the situation and action vectors, but nothing concerning the Q-value. That is,

$$d\,(i) = |situation - w_{situation,i}| + |effective\_action - w_{action,i}|.$$

The learning algorithm updates the Q-value weight and, the situation and action weights. Together with the selected neuron, the neighbors are also updated. During the learning, the influence on the neighbors decreases inversely proportionally to the number of iterations. The properties (local representation of the probability densities) of the self-organizing map allow one to predict that, if a correct behavior is learned (i.e., only positive rewards are experienced), then all neurons will code positive Q-values.

In our experiments, there were 16 neurons, a neighborhood of 4 and 18 inputs (16 IR sensor values, 1 action value, 1 Qvalue). Figure 9 shows the evolution of #r+/it. and #r-/it. during the experiments.

During the initial 200 Q-learning iterations, we can see the evolution of #r+/it. and #r-/it.. The remaining 100 iterations are used to verify the level of performance (test phase): positive rewards stable at about 22% and negative rewards converging towards 0%.

Figure 10 displays an “objective” measure of the performance of the behavior: The distance to the obstacles. The only combination allowing a distance to the obstacles greater than a random move selection behavior corresponds to the threshold configuration given by the UPA, meaning that the discovered RF parameters (θ_+ and θ_-) are accurate for this task. There are 200 learning iterations, after which there is no more learning for the last 100 iterations. We are able to verify that the optimally parametrized RF allows the learning of a behavior that avoids obstacles from a greater distance.

Fig. 10. Distance to the obstacles vs. number of obstacles encountered during the 300 learning iterations. (a) is obtained by a random move selection behavior (no learning involved), (b) is the learned behavior corresponding to the values given by UPA ($\theta_+ = 390$, $\theta_- = 1150$), (c) corresponds to ($\theta_+ = 195$, $\theta_- = 2300$), (d) corresponds to ($\theta_+ = 780$, $\theta_- = 575$), (e) corresponds to ($\theta_+ = 195$, $\theta_- = 1150$), (f) corresponds to ($\theta_+ = 780$, $\theta_- = 1150$), (h) corresponds to ($\theta_+ = 390$, $\theta_- = 2300$), (i) corresponds to ($\theta_+ = 390$, $\theta_- = 575$).

6 Discussion

The role of UPA, as described in this paper, is to guarantee an appropriate proportion of positive and negative reinforcements during the initial "random" exploration phase. However, once RL is activated, the proportions will change as learning progresses. Could it be useful to combine UPA and RL?

There are primarily two kinds of exploration techniques. The directed exploration techniques [8] emphasize the fact that the robot extensively visits the environment. On the other hand, the non-directed exploration techniques are based on strategies of visiting states (or state-action pairs) around previously rewarded ones. In general, the strategies have an exploratory component [8,9,12], which decays with time, and in all cases they do not take into account the evolution of the robot performance.

Using UPA during learning could help control the probability of obtaining positive and negative rewards. The distribution of situations will change during the experiment, due to the exploitation of the Q-learned knowledge. This will affect the adequacy of the UPA parameters (computed during exploration). UPA will have to dynamically adjust the values of the ratios during the learning process to be consistent with the learned knowledge. If ratios of 0.2 appear logical for a random exploration, they are not for a learned policy. In this case, we would prefer ratios of values +1 and 0 (for p_+ and p_- resp.). Therefore, UPA will have to update the goal ratios during the learning. How to measure the quality of the acquired knowledge so as to be able to correctly fix the goal ratios is a difficult problem. We propose to use the monotonicity of the RPR and allow only increase in θ_+ and θ_-, stopping the update of θ_i as soon as the performance ratios of the robot do not follow the desired ones. The desired performance starts at 0.2 for p_+ and increases towards +1, where it starts at 0.2 for p_- and decreases towards 0. The implicit idea is that the value of θ_+ should be as small as possible and the value of θ_- should be as large as possible; otherwise a trivial solution is $\theta_+ = \theta_- = 0$.

7 Conclusion

In this paper, we have proposed to solve, for a particular case, the so-called exploration/exploitation dilema. Based on the assumption that the reinforcement function can be expressed as a constraint involving only one threshold parameter per case ($\theta_+ \to \#r_+$, $\theta_- \to \#r_-$), the proposed Update Parameter Algorithm (UPA) allows one to compute during a random exploration an estimation of the RF threshold values. We test the tuned values using a mobile Nomad 200 robot on a task of synthesis of an obstacle avoidance behavior. The performance of the learning was evaluated along two indexes: Ratio of positive and negative rewards over time ($\#r_+/it.$ and $\#r_-/it.$) and distance to the obstacles. Self-organizing maps were used to implement the RL. Several experiments have been made with values of the parameters in the RF definition both close to or far from those obtained with the UPA. The results showed the validity of the RF thresholds obtained, demonstrating that RF design is possible and efficient. Evidently, there are some restrictions on the use of UPA. The coupling of UPA and RL may generate some important gains in convergence ability and speed. It is our desire to address this issue in the near future.

Acknowledgements

The authors would like to thank L. Parker (CESAR-ORNL) and the anonymous reviewers for helpful comments, suggestions and discussions.

References

[1] D. Ackley and M. Littman, Interactions Between Learning and Evolution, in By C.G. Langton, C. Taylor, J. D. Farmer & S. Rasmussen eds., Artificial Life II,

SFI Studies in the Sciences of Complexity, vol. X, Addison Wesley Publishers, (1991) 487-509.

[2] P. Dayan and T. Sejnowski, TD(l) Convergence with Probability 1, Machine Learning 14(3) (1994).

[3] L.P. Kaelbling, M.L. Littman and A.W. Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4, (1996) 237-285.

[4] L.-J. Lin and T. M. Mitchell, Memory Approaches to Reinforcement Learning in Non-Markovian Domains, Technical Report CMU-CS-92-138, Carnegie Mellon University, School of Computer Science, 1992.

[5] S. Mahadevan and J. Connell, Automatic Programming of Behavior-based Robots using Reinforcement Learning, Artificial Intelligence 55 (1992) 311-365.

[6] M. J. Mataric, Reward Functions for Accelerated Learning, in W.W. Cohen and H. Hirsh, eds. Machine Learning: Proceedings of the Eleventh International Conference, Morgan Kaufmann Publishers, (1994) 181-189.

[7] S. Sehad and C. Touzet, Reinforcement Learning and Neural Reinforcement Learning, Proc. ESANN'94, (1994).

[8] S.B. Thrun, Efficient Exploration In Reinforcement Learning, Technical Report CMU-CS-92-102 Carnegie-Mellon University, 1992.

[9] S. Thrun, Exploration in Active Learning, in: M. Arbib ed., Handbook of Brain and Neural Networks, (1996).

[10] C. Touzet, Neural Reinforcement Learning for Behaviour Synthesis, Robotics and Autonomous Systems 22, Special issue on Learning Robot: the New Wave, N. Sharkey Guest Editor, (1997) 251-281.

[11] C. Touzet and S. Sen, Learning Agents, invited paper at the First Conf. on Autonomous Agents, Marina del Rey, CA, (1997).

[12] P. Zhang and S. Canu, Indirect Adaptive Exploration in Entropy-based Reinforcement Learning, Proc. of ICANN'96, (1996).

Since 1993, Juan Miguel Santos has been an Assistant Professor at the Departamento de Computacin of the University of Buenos Aires (Argentina). He received his “Licenciatura en Ciencias de la Computacin” (MSc) in 1993 and his Ph.D. degree in Computer Sciences (jointly with the University of Marseilles, France) in 1999 from the same University. His research interests are in reinforcement learning, autonomous robotics and artificial neural networks.

Claude Touzet is a visiting scientist at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). He received his M. Sc. degree in Behavioral Neurosciences in 1985, his Ph.D. degree in Computer Sciences in 1990 from the University of Montpellier (France) on the subject of sequential neural networks, and his “Habilitation à Diriger des Recherches” in 1998 from the University of Marseilles (France). He was a researcher associated with the LERI-EERIE from 1987 to 1994, and an Associate Professor at the University of Marseilles from 1994 to 1997, where he conducted is research at the DIAM-IUSPIM. His research interests are in cooperative robotics, autonomous robotics, reinforcement learning and artificial neural networks. He participates in several projects involving artificial neural networks and robotics with companies or agencies like DARPA, Renault, Usinor-Sacilor, French Ministry of Defense, K-Team SA, etc.