Title: Contribution to the study and the design of reinforcement functions

Author: Juan Miguel Santos

e-mail: jmsantos@dc.uba.ar

phone: 54 11 4576-3390 to 93 extension 708

fax: 54 11 4576-3359

Directors: Dr. Hugo Daniel Scolnik (Universidad de Buenos Aires)

Dr. Norbert Giambiasi (Universite d'Aix-Marseille III, Francia)

Advisor: Dr. Claude Touzet (CESAR, Oak Ridge National Laboratory, TN, EEUU)

University: Universidad de Buenos Aires (*) (Facultad de Ciencias Exactas y Naturales, Departamento de Computación)

(*) Jointly with Universite d'Aix-Marseille III under tutelary agreement.

We have studied the Reinforcement Function Design Process in two steps. For the first one we have considered the translation of a natural language description into an instance of our proposed Reinforcement Function General Expression. For the second step, we have gone deeply into the tuning of the parameters in this expression. It allowed us to obtain optimal definitions of the reinforcement function (relative to exploration). Since the General Expression is based on constraints, we have indentified them according to the type of state variable estimator on which they act, in particular: position and velocity.

Using a particular, but representative Reinforcement Function (RF) expression, we study the relation between the Sum of each reinforcement type and the RF parameters during the exploration phase of the learning. For linear relations, we propose an analytic method to obtain the RF parameters values (no experimentation requires). For non-linear, but monotonous relations, we propose the Update Parameter Algorithm (UPA) and show that UPA can efficiently adjust the proportion of negative and positive reinforcements received during the exploratory phase of the learning.

Additionally, we study the feasibility and consequences of adapting the RF during the learning process so as to improve the learning convergence of the system. Dynamic-UPA allows the whole learning process to maintain a desired ratio of positive and negative rewards. Thus, we introduce an approach to undertake the exploration-exploitation dilemma - a necessary step for efficient Reinforcement Learning.

We illustrate, with several experiments involving robots (mobile and arm), the performance of the proposed design methods. Finally, we emphasize the main conclusions and present some future directions of research.

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

Contribution to the study and the design of reinforcement functions

Juan Miguel Santos

(#) Departamento de Computación, F.C.E.yN., Universidad de Buenos Aires;

Ciudad Universitaria, Pabellón I; 1428 Bs. As., Argentina. Email: jmsantos@dc.uba.ar

Abstract. We have studied the Reinforcement Function Design Process in two steps. For the first one we have considered the translation of a natural language description into an instance of our proposed Reinforcement Function General Expression. For the second step, we have gone deeply into the tuning of the parameters in this expression. It allowed us to obtain optimal definitions of the reinforcement function (relative to exploration). Since the General Expression is based on constraints, we have indentified them according to the type of state variable estimator on which they act, in particular: position and velocity.
Using a particular, but representative Reinforcement Function (RF) expression, we study the relation between the Sum of each reinforcement type and the RF parameters during the exploration phase of the learning. For linear relations, we propose an analytic method to obtain the RF parameters values (no experimentation requires). For non-linear, but monotonous relations, we propose the Update Parameter Algorithm (UPA) and show that UPA can efficiently adjust the proportion of negative and positive reinforcements received during the exploratory phase of the learning.
Additionally, we study the feasibility and consequences of adapting the RF during the learning process so as to improve the learning convergence of the system. Dynamic-UPA allows the whole learning process to maintain a desired ratio of positive and negative rewards. Thus, we introduce an approach to undertake the exploration-exploitation dilemma - a necessary step for efficient Reinforcement Learning. We show, with several experiments involving robots (mobile and arm), the performance of the proposed design methods. Finally, we emphasize the main conclusions and present some future directions of research.

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

1. Introduction

The objective of Reinforcement Learning (RL) is to synthesize a mapping between states and actions that maximizes rewards over time. Dynamic Programming deals with some of the problems related to this type of learning, mainly by the use of two algorithms (or variations of them): Policy and Value Iteration. However, these algorithms require the problem to be modeled as a Markovian Decision Process. This constraint does not apply to a large set of applications and real cases. Sutton [24] introduces a careful study of the Temporal Differences method: if the world model is unknown, this method allows us to deal with the *temporal credit assignment* problem. That is, how to assign responsibilities to the actions carried out in the past for current delayed reinforcements. He proposes the Adaptive Heuristic Critic (AHC), which does not require an explicit knowledge of the Transition Function of the system. One year later, Watkins [33] introduced Q-Learning which has one major advantage over Sutton's proposal: the Evaluation and Policy functions are joined together in a Q-function. Convergence of Q-learning has been studied in [34].

Despite these important contributions, the necessity of a finite state-action space represents a serious obstacle to real RL applications. Most of the time, it is not feasible to use these techniques in applications whose state space is too large to be computationally treated. Therefore, the learning scheme must provide a methodology to *memorize* and *generalize* the gathered knowledge. Generalization and memorization processes are intimately connected. Generalization is the ability to associate a legitimate output with an unforeseen input. The generalization process must not only allow the system to decide what action to perform in an unknown state, but it must also allow the updating rule to generalize with respect to the current state-action pair. This problem is known as *structural credit assignment* [12]. The *exploration process* is also a difficult task when dealing with huge state-action spaces. Since the robot cannot visit the whole set of states, it must have some strategy to select the relevant ones. This strategy involves the generalization methodology. The information obtained for one state-action pair may be generalized to another's, allowing the exploration to focus on other particular regions of the search space. During the last decade, a number of contributions on memorization, generalization [2,3,8,9,10,11,22,30,31] and exploration [28,29,35] have been made, allowing the use RL in real applications.

However, efficient memory, generalization and exploration are powerless against an inefficient RF definition. Q-Learning, AHC and variants of them search to maximize a measure of performance[1]. In all cases, this measure depends of the reinforcements received by the agent during learning. Each learning technique uses some function (i.e. Q-function, Valuation function) to ponder each possible action and choose the best one. Usually, this function expresses the expectation of the performance measure given a state or state-action pair. Since the RF has the responsibility of delivering the reinforcements during learning, its definition becomes a critical and delicate task for the RL scheme designer. Pure delay problems (e. g., backgammon [25], "car in the hill" [9], T-maze problems, etc.) which are associated with a unique reinforcement value corresponding to the goal state are the exception. Usually, real applications with continuous output sensors present serious difficulties for defining the adequate RF definition. A RF can be considered defective if it does not allow the synthesis of the desired behavior. There are two possible reasons why this can happen: (1) the time requested to achieve convergence of the learning is prohibitive and/or (2) the synthesized behavior is not the desired one. Our objective is to provide the operator with algorithms that optimize the learning speed so has to be able to reduce the type of failures to the second case only – the RF synthesizing the wrong behavior.

Some authors have explored indirectly the idea of the RF definition. Their works have been mainly based on the use of some measure of performance to evaluate the goodness of a RF. For example, Schmidhuber et al. [23], use as a criterion acceptance (or rejection) of previously imposed changes on a Policy, the evolution of the Reinforcements Sum received from the last change. Mataric [15,16], and Millán [17] use progress estimators to measure the evolution of the system in achieving sub-objectives (for simple behaviors with an associated metric) expressed in the RF. On the other hand, Ackley and Littman [1] introduce an Evolutionary Reinforcement Learning scheme where RL allows individuals to improve their performance along their life. Thus, the performance of the complete system is indirectly used for accepting, or not, changes of the RF definition of each individual. However, it is our opinion that these attempts failed to achieve a rigorous development in regards to the exploration-exploitation dilemma, and its related issues: learning speed, convergence, etc.

2. From behavior specification to Reinforcement Function definition

A behavior is a mapping between sensory states and actions. It could be represented in several ways, and we want to distinguish three of them:

- The first one, and maybe the more usual, is a description in a natural language. This expression has a high semantic content but it is not directly usable to program agents (or robots).
- A second kind of representation is the codification in terms of what the agent is able to perceive and of how it should act in consequence. This way to represent behaviors is directly usable to program agents (or robots).
- The third kind of description comprises: to rewarding situation-action pairs that are part of the desired behavior and punishing those that must be excluded.

Figure 1 sketches out a scheme where three methodologies to obtain behaviors are presented. The left arrow goes from the Natural Language description to the mapping one. This methodology requires the human operator to interpret the semantic of the description and code it in terms of a program. In the middle of the figure is represented the Supervised Learning strategy. In this case, the human must use the behavior description to generate a representative learning base. Then, a supervised learning technique synthesizes the desired mapping. But, it is not always easy to generate a learning base that is representative. It represents an important effort of the human operator if it is achievable. Finally, on right of the figure, the RL approach is represented. In this case the human task is to obtain a measure of performance that indicates in qualitative way how good the behavior learned by the agent is, relative to a desired behavior. It is achieved by finding a partition set of the state-action space using a set of constraints. According to the type of constraints, several estimators (if necessary) of the state variables are used in the Reinforcement Function definition. Then, a RL technique generates the mapping corresponding to the RF definition. Note that, if the description of the designer's knowledge of how to specify the performance measure is inexact or incomplete, the learning technique will allow the system to evolve by maximizing the sum of reinforcements, but the obtained behavior will not accomplish the desired task.

There is no doubt that the Reinforcement Function definitions constitute the weakest part of the RL. There are problems where the definition of the RF can be achieved in an *immediate* way. For example, the "two arms problem" has the definition of the RF in its description. Also, there are problems where the definition can be achieved in *simple* way and it is enough for the learning technique to reach the desired behavior (i.e., pure delay reward problems such as T-maze, or even in some board games). However, these cases are strongly based on the ability of the technique to deal with the credit assignment and structural credit assignment problems.

[1] Typically, the Discount Reinforcement Infinite Sum is used due to its useful convergence property.

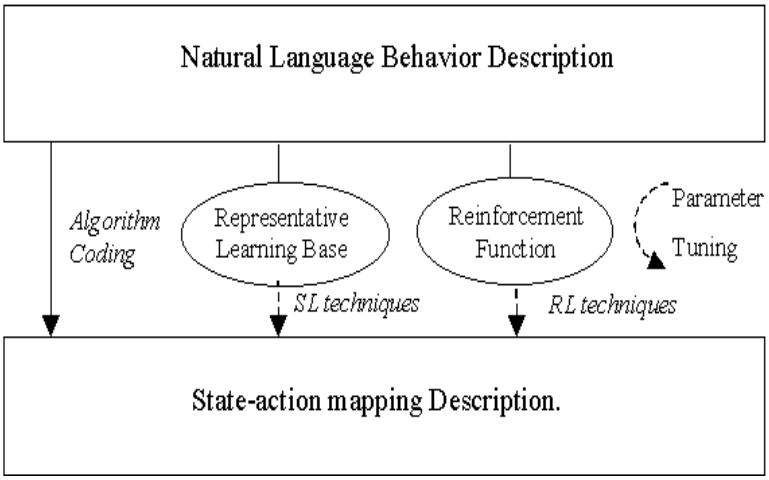

Figure 1. Three methodologies to obtain the mapping (behavior). Solid lines represent human effort to translate the semantic information of the description into a formal one. Dash lines represent automatic techniques (learning).

In spite of these cases, there are problems where the RF definition cannot be achieved either in an immediate nor a simple way. A designer must try several RF definitions until finding an acceptable one. What methodology could a designer of a RL scheme use in order to define the right RF? We will point out some aspects of the RF design and we will present some tools that may be of help to the designer.
The RF definition process includes three steps (see figure 2):

1) Define a set of constraints,
2) Establish the logical relations among them,
3) Tune (if necessary) the parameters associated with each constraint.

Basically, the RF divides the set formed by the possible situations and actions in (a few) classes (partition). Typically, these classes are associated with positive, negative and null rewards. Each subset inside the partition is defined by one or more *constraints*. They represent the designer knowledge about the desired behavior. Constraints may take into account not only the characteristics of the behavior itself, but also, the dynamics of the agent during the behavior and some specific aspects of the environment such as the shape (or some particular identification) of certain objects, location of certain points, etc. The designer must identify those elementary constraints and relate them by means of the use of logical operators. Finally, if an elementary constraint depends on some kind of parameter, the designer adjusts its value in order to complete the semantics of the RF.

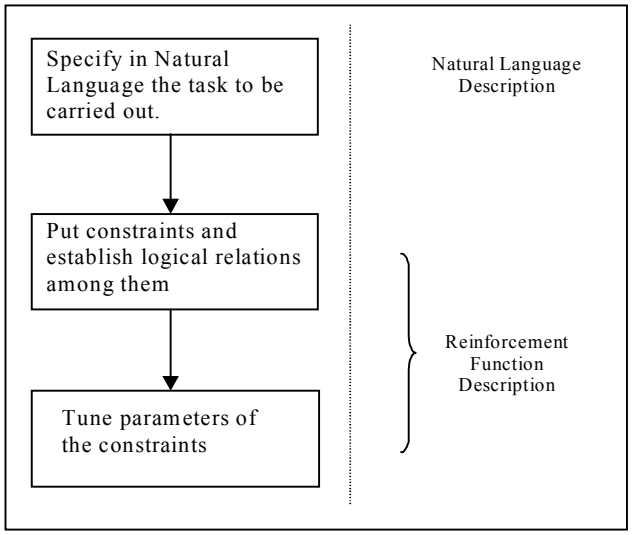

Figure 2. Scheme of the RF definition process. Right, types of description used in each step of the process.

2.1 Reinforcement Function expression

There is no standard for the specification of the RF in the literature. Usually, each author expresses the definition in his/her own way without much explanation. In some cases, the details of how this function was implemented are not even communicated. We propose a general expression of the RF - general in the sense that any behavior can be synthesized using an instantiation of this expression. This general expression allows us to study some of the major characteristics of the RF, and allows us to propose an automatic tuning of the RF definition. The general

expression is given by:

$$(1)\qquad RF\left((s_1,..s_u,a)^t,..,(s_1,..s_u,a)^0\right)=\begin{cases} prior(c_+,c_-) & if \quad (c_+ \cup c_-) \quad is \quad true \\ 0 & otherwise \end{cases}$$

where $(s_1,..,s_u,a)^t$ is the sensor and action values vector at time *t*, and *u* is the number of agent sensors, c_+ is a list of constraints related with logical operators defined by $c_+ = [[not]e_1 \ \ and \mid or \ \ [not]e_2 \ \ and \mid or..[not]e_{\#c_+}]$ where each expression $e_w:(g_k \ \ oper \ \ \theta_j)$, $w:1..\#c_+$, is a relation (inequality or equality) between one function $g_k(\ )$ and one parameter $\theta_j; j:1..q$, with $oper \in \{>,<,\geq,\leq,=\}$, c_- is defined in similar way to c_+ and *prior* is a predicate, which evaluates the lists of constraints, setting the priorities between them and gives one value of reinforcement, $prior:\{c_+,c_-\}\rightarrow\{+1,-1\}$. (In order to improve the understanding of the general expression, the reader can find several examples of instantiation in [21].)

This general expression allows taking into account past states and past actions. We would like to emphasize that this does not intend to ignore the severe limitations that the RL has in order to deal with non-Markovian Decision Problems. Simply, there are an important number of cases where the environment is partially observable and the true state maybe reconstructed using information about the evolution of observable states.

2.2 Specifying constraints in the Reinforcement Function

The aim of constructing constraints based on elementary ones is to allow the instantiation of the general expression in a particular definition of the RF. From this point of view, it is useful to characterize some of the elementary constraints according to the type of the state variable on which they operate. Thus, we distinguished two types of functions $g(\ )$: *position estimators* and *velocity estimators*. In both cases, the obtained estimations will be relative to the agent's environment and their precision will depend on the nature of the sensors used.

Typically, the sensors give an estimate of the distance between an agent and an object, and this could be used like a measure of their relative position in the environment. However, the concept of position could vary according the sensory space. For example, if the sensors perceive temperature in a three dimensional space, the position would be an estimation of the field function ($F:\Re^3\rightarrow\Re$) value. Anyway, here we will refer to sensors that give a distance estimation, since the generalization to other types of sensors can be made straightforwardly.

2.2.1 Estimators of position in the sensory space

We define the position estimators using two approximations:

1) If the sensors measure the area free of objects around the agent, the integration of the perceived areas of each sensor gives a measure of the average distance of the agent with respect to its closer environment. We call *integral estimator* the function $g(\ )$ that implements this approach. An approximation could be defined by $\Im_I = \sum_{i\in Sensor\_index\_set} \rho_i S_i$, where the *Sensor_index_set* contains the sensor indexes involved in this estimation, and ρ_i is a constant associated to each sensor (related with its sensitivity and coverage area).

2) Instead of the integration of the perceived areas, the distance to the nearest object S_m could be obtained by means of a *maximum (or minimum) estimator* $\Im_m$ given by $\Im_m = \min_{i\in Sensor\_index\_Set}\{\rho_i s_i\}$, where *Sensor_index_set* and ρ_i have the same meaning used as $\Im_I$.

2.2.2 Estimators of position change

The positions estimators are useful to define constraints on the relative position of the agent with respect to an object, but they are insufficient to define constraints on the agent dynamics. Therefore, it is mandatory that the RF has a way of estimating the speed at which it is avoiding the obstacles. We will consider two types of velocity estimation (in both cases, we propose to use an approximation of the derivatives of S_I and S_m in order to obtain such an estimate):

1) The change of the total perceived area by the sensors over time (S_I'): let $S_I(t)$ and $S_I(t+1)$ be the values of the integral estimator at time *t* and *t+1* respectively, then the velocity of change of S_I (or *diferential*

estimator) can be approximated by

$$\tilde{S}'_I = \frac{\Delta S_I}{\Delta t} = \frac{S_I(t+1) - S_I(t)}{\Delta t}.$$

If $\tilde{S}'_I > 0$, it implies that the agent moved away from the object (or objects) respectively to its previous position at *t*. For example, $\tilde{S}'_I > \varepsilon$ ($\varepsilon > 0$) could be an adequate constraint for positive reward if one wants to reward escaping obstacles.

2) The change of the minimum (maximum) perceived area by the sensors over time (S'_m): let $S_m(t)$ and $S_m(t+1)$ be estimations of the distance from the agent to the closer object at time *t* and *t+1* respectively; then the rate of change of S_m can be approximated by $\tilde{S}'_m = \frac{\Delta S_m}{\Delta t} = \frac{S_m(t+1) - S_m(t)}{\Delta t}$.

$\tilde{S}'_m > 0$ implies that the agent moved away from some object/s that affect a particular region in its surrounding.

2.3 Tuning Reinforcement Function parameters

Once the elementary constraints have been identified, the value of the parameters associated with each elementary constraint must be computed. This steps represents an important consumption of time for the designer because, this is carried out by means of trial and error. Thus, a parameter tuning technique is especially useful here. The initial assumption is based on the existence of a relationship among the values of the parameters and the proportion of positive, negative or null reinforcements during the exploration phase of the learning. Our hypothesis is that there are values of the RF parameters which guarantee a proportion of each type of reinforcement during the random exploration.

2.3.1 Monotony between RF parameters and reinforcements

We observe a monotony relation between the number of positive (negative) reinforcements and the values of θ^+ and (θ^-). Let us consider the following reinforcement function:

$$RF(s) = \begin{cases} +1 & if \quad g_1(s) < \theta^+ \\ -1 & if \quad g_2(s) < \theta^- \\ 0 & otherwise \end{cases}$$

Definition and monotony proof: We define the number of positive reinforcements as $\#r^+(\theta^+) = \sum_{i=0}^{t} r_i(\theta^+)$, where

$$r_i(\theta) = \begin{cases} 1 & g(s^i, a^i) < \theta \\ 0 & else \end{cases}.$$

In order to demonstrate that $\#r^+$ and θ^+ suffer a weak increasing monotony it is sufficient to see that with any $\theta' > \theta^+$ there cannot be less positive rewards ($g(\bar{s}_i) < \theta^+ < \theta'$) independent on the characteristics of g. If we have more than one parameter θ then changing one of them holds up with the same monotony (same proof for $\#r^-(\theta^-)$ and θ^-).
In the case the RF is defined with ">" the relation is a weak decreasing monotony (same proof). Anyway, we usually generalize it to the increasing case - due to the fact that we can change the constraint parameter and function: $g(.) > \theta \rightarrow g'(.) < \theta'$, where $g' := -g$ and $\theta' = -\theta$.

Based on this monotony relation, it is possible to try the search of θ^+ and θ^- that guarantees the best approximation of the given number of positive and negative reinforcements, respectively.
During the initial learning phase, the agent must intensively explore its environment. *Therefore, it is reasonable that the agent has a balanced proportion of negative and positive rewards so as to generate a representative sampling of the state-action pairs corresponding to each type of reinforcement.*

2.3.2 Special monoty case: linear relation.

Let us consider now, the class of RF where the positive rewards and negative rewards are each associated, with a

simple constraint

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

Under the condition of uniform distribution during exploration, the relation between reinforcements and parameters of the RF can be modeled by means of the knowledge of the estimators $g_1(\ )$ and $g_2(\ )$. If such estimators are linear, we can express the sum of positive (and negative) reinforcements as

(1) $\#r_+ = m\theta_+ + b$, and

(2) $\#r_- = m'\theta_- + b'$,

where *m, b, m'* y *b'* depend of the $g_1(\ )$ and $g_2(\ )$ inverses and other physics constants (see [21] for details). Equations 1 and 2 show the strong dependency of ($\#r_+$ vs. θ_+) and ($\#r_-$ vs. θ_-) with the functions $g_1(\ )$ and $g_2(\ )$. This is a particular case of monotony in Reinforcement Parameter Relation given by the linear relation.

Let us call p_+, p_- and p_0 the desired values (ideal ratios) for the observers $\frac{\#r_+}{t}$, $\frac{\#r_-}{t}$ and $\frac{\#r_0}{t}$. These observers indicate the proportion of reinforcements resulting from the current value of the parameters in the RF.

Now, we consider the following function $z(t)=\frac{\#r_-(t)}{\#r_+(t)}$. During the exploration part of the learning phase, $z(t)$ should be as close as possible to $\frac{p_-}{p_+}$. This means: $\frac{\#r_-(t)}{\#r_+(t)} \to \frac{p_-}{p_+}$.

Therefore, using equations 1 and 1 and assuming that the ideal ratios are reached, we have $\frac{p_-}{p_+}=\frac{\theta_- m'+b'}{\theta_+ m+b}$, or

(3) $$\frac{p_-}{p_+}(b-b')=\theta_- m'-\frac{p_-}{p_+}\theta_+ m .$$

On the other hand, we have

(4) $maxIt = \theta_- m'+b'+\theta_+ m+\#r_0$

where $\#r_0 = maxIt.(1-p_+-p_-)$ and *maxIt* is the maximum number of iteration for the exploration phase.

Solving the system formed by equations 3 and 4, we obtain that

$$\theta_- = \frac{-b'+maxIt.p_-}{m'a}, \text{ and } \theta_+ = \frac{maxIt.p_+(p_+-p_-)}{(p_+-p_-)}-\frac{b}{m}.$$

Summing up, we have described here a method to compute, before the learning phase, the parameters used in the RF under the assumptions that the functions $g_1(\ )$ and $g_2(\ )$ are linear and the distribution, during the exploration phase, is uniform. The values of *m* and *b* are obtained analytically.

2.3.3 Update Parameter Algorithm (UPA)

Let us consider again the following class of RF

$$RF(s)=\begin{cases}+1\ if(g_1(s) \quad oper_+ \quad \theta_+)\\ -1\ if(g_2(s) \quad oper_- \quad \theta_-),\\ 0 \quad otherwise\end{cases}$$

(where the situation is represented by the sensor vector *s* and $(g_1(s) \quad oper_+ \quad \theta_+)$ and $(g_2(s) \quad oper_- \quad \theta_-)$ are two constraints parameterized by θ_+ and θ_-) that has associated two important relations between the sum of received reinforcements and the parameter values: the sum of positive reinforcements ($\#r_+$) depends on the value θ_+ and the sum of negative reinforcements ($\#r_-$) on θ_-. Based on the monotonicity property of these relations, we have proposed an algorithm to dynamically compute, during a pure random exploration phase, the parameter values. The principle of the Update Parameters Algorithm (UPA) is described in figure 3.

Let us call the variables p_+ and p_- the desired values or ideal ratios ($0< p_+, p_- <1$) 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 numbers of positive and negative rewards occurring in the last Δt iterations ($\#r_+^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, 1)$, where $\delta(x,y) = \begin{cases} 1 \; if \; x = y \\ 0 \; otherwise \end{cases}$) (resp. negative rewards occurring in the last Δt iterations $\#r_-^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, -1)$). 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 next iterations a value of $\frac{\#r_+^{\Delta t}}{\Delta t}$ closer to the desired value p_+. If the relation between $\#r_+$ and θ_+ is monotonically decreasing, the value of $\Delta\theta_+$ is added to θ_+. In the opposite case, $\frac{\#r_+^{\Delta t}}{\Delta t}$ is smaller than p_+, then the value $\Delta\theta_+$ is substracted from θ_+. If, instead, the relation is monotonically increasing the sign of $\Delta\theta_+$ must be inverted. (First part of step 3 of UPA).

Initialization:
Define ε, *k,* k_1 *and* k_2 $(0 < k_1, k_2 < 1)$,
Define the values of p_+ *and* p_-,
Choose some initial value for θ_+ *and* θ_-,
Set the initial step for $\Delta\theta_+$ *and* $\Delta\theta_-$,

1. Compute $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ as

$$\#r_+^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, 1); \qquad \#r_-^{\Delta t} = \sum_{i=t-\Delta t}^{t} \delta(r^i, -1)$$

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 Δr_+ and Δr_- the current values of the observer estimators,

$$\Delta r_+ = \frac{\#r_+^{\Delta t}}{\Delta t} \; ; \; \Delta r_- = \frac{\#r_-^{\Delta t}}{\Delta t}$$

5. If during *k* iterations

$$\left|\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right| < \varepsilon \text{ and } \left|\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right| < \varepsilon \text{; (where } \varepsilon > 0 \text{ and small)}$$

then stop
else go to 1.

Figure 3. Update Parameter Algorithm.

If the value $\frac{\#r_-^{\Delta t}}{\Delta t}$ is greater than p_-, then we have to modify θ_- to obtain in the next iterations a value of $\frac{\#r_-^{\Delta t}}{\Delta t}$ closer to the desired value p_-. If the relation between $\#r_-$ and θ_- is monotonically decreasing, 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 substracted from θ_-. If, instead, the relation is monotonically increasing the sign of $\Delta\theta_-$ must be inverted. (Second part of

step 3 of UPA).

If UPA crosses the ideal value of θ_+ (and θ_-) the substraction between the previous estimation of the real observation and p_+ (and p_- resp.) will change the sign. Then, the product $(\Delta r_+ - p_+)\left(\frac{\#r_+^{\Delta t}}{\Delta t} - p_+\right)$ is negative ($(\Delta r_- - p_-)\left(\frac{\#r_-^{\Delta t}}{\Delta t} - p_-\right)$ resp.). To this end, 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 $\Delta\theta_+$ (resp. $\Delta\theta_-$) 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).

UPA as presented in this section, its is independent of the learning technique. UPA works by, during a pure random exploration phase, tuning the parameters of the RF to obtain a RF definition that guarantees a desired proportion of positive and negative reinforcements during the initial exploration part of the learning.

3. Tuning of the parameters during learning: Dynamic-UPA

Once the Q-learning is activated, the proportions of negative and positive reinforcements will change as the learning progresses. Dynamic-UPA is the simultaneous use of UPA and Q-learning. It controls the probability of obtaining rewards and penalties during exploration using an evaluation of the performance measure. For this purpose, it is necessary that the ideal proportions p_+ and p_- convergence towards the expected values representative of a completed learning process. That is, p_+ close to 1 and p_- close to 0.

In standard approaches, if the probability of obtaining a positive reinforcement is fixed and small, the probability of exploring successive states from one that had an unfortunate election of actions during the exploration (with associated penalties) will also be small. Therefore, the learning process will be unnecessarily delayed. Instead, the way in which Dynamic-UPA facilitates or not the exploration of new states does not depend exclusively on time or probability decaying along time, but also on the capacity of the learning to improve performance. The probability of rewards and penalties are controlled by changing the size of the subsets in the RF partition (positive and negative subsets). If $\frac{\#r+}{t}$ is less than p_+, then Dynamic-UPA produces an *expansion* (increases the size of the positive subsets). If instead, $\frac{\#r+}{t}$ is greater than p_+, then Dynamic-UPA produces a *contraction* (decreases the size of the positive subsets).

At the beginning of learning most of the rewards come from exploration. To achieve $\frac{\#r+}{t}$ approaching p_+, Dynamic-UPA must produce an expansion of the positive sets. An analogous situation occurs with the negative reinforcement. As the learning progresses, the exploitation of previously rewarded pairs (during the exploration) produce an increase in $\frac{\#r+}{t}$. Therefore, Dynamic-UPA changes the value of the RF parameters to contract the positive subsets. Thus, Dynamic-UPA controls, by means of the expansion, the exploration of pairs that lead to states close to a solution and, by means of the contraction, to select those that are closer to it.

Let $\frac{r}{n}$ be an estimation of $\frac{\#r+}{t}$, where r is the number of positive rewards gathered during the last n iterations If, during the next iteration n+1, a reward is obtained, the new estimation of the observer will be $\frac{r+1}{n+1}$. Instead, if a null or negative reinforcement is obtained, the new estimation becomes $\frac{r}{n+1}$. The former will produce a contraction while that the latter will produce an expansion. Now, we want to know how the observer will be

affected. Let $\left(\frac{r+1}{n+1}-\frac{r}{n}\right)$ be the magnitude due to contraction and $\left(\frac{r}{n}-\frac{r}{n+1}\right)$ be the magnitude due to expansion. We can express the subtraction between them as $\frac{n-2r}{n(n+1)}$. Since, $\frac{\#r+}{t}$ is close to p_+, then *2r>n*. Therefore, the expansion effect is more important than the contraction one. That is, null or negative reinforcements produce a more significant change than the rewards. Thus, if the estimator is in a damped oscillation, then the learning process is in a convergence phase (systematic delivery of rewards due mainly to exploitation). If instead, the estimator is in a non-damped oscillation, then the learning process is mainly in a exploration phase.

We denote the ideal proportions of p_+ and p_- as p_{+ideal} and p_{-ideal} respectively. p_{+ideal} cannot be equal to 1 due to the exploratory component during learning. Although the agent has learned a good policy (in this case, we could expect $\frac{\#r+}{t}$ trends to 1), the exploratory component forces the robot to try new actions and could lead to null or negative reinforcements. Similarity, we could not expect $\frac{\#r-}{t}$ is equal to 0. The values of p_{+ideal} and p_{-ideal} are related to the difficulty of learning a given behavior. In fact, the use of UPA before learning can give clues to the designer of the RL scheme about this difficulty .

4. Experiments

We have conducted a series of experiments with a robot arm and a mobile robot. In each case, we have used Reinforcement Function definitions with the aim of emphasizing diverse aspects that have been studied in this thesis. In all the cases, the memorization and generalization strategies use RBF artificial neural networks [19]. This strategy slightly differs of previous approaches using RBF networks [2,9] or neural networks with radial basis units in their hidden layer [17]. The distribution of inputs and outputs were defined in order to facilitate the use of the RBF network as an associative memory. This difference with respect to traditional Q-function representation (situation and action as inputs, Q-value as output) allowed us to implement the procedures for retrieving best associated action and Q-max (in both cases for a given situation) which are explained in the next section.

4.1 Neural implementation

Figure 4 shows the network used for the representation of the Q-function.

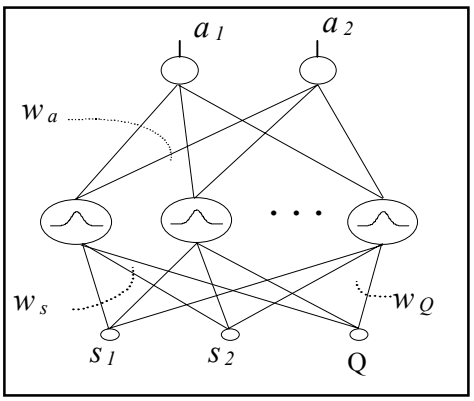

Figure 4. Architecture of the RBF network. The number of hidden units is modified during the learning phase according to the situations visited. A hidden unit is added when a new situation does not belong to any of the existing receptive fields of the hidden units. The output $a_1 .. a_n$ are the components of the action vector *a*. The inputs $s_1 .. s_m$ are the components of the situation vector *s*. *Q* is the Q-value associated with the pair (*s,a*).

The whole strategy includes a growing algorithm used to obtain the number of hidden units, which allows a precise representation of the situation-action pair space. Figure 5 describes the use of the RBF network as an associative memory to obtain the value of the action *a* associated with a situation *s*. As we are looking for the cluster with higher associated Q-value, the input to this algorithm will be *s* and 1 respectively. If a *new* hidden unit must be added, then its connection weights with the inputs will be initialized using the situation vector *s*; the Q-value component will be initialized to zero. The connection weight corresponding to the action will be initialized with a random value between 0 and 1 (normalized value). Figure 6 describes the weight-updating rule for the conducted action *a* starting from the situation *s*, receiving the reinforcement *r* and the Q-max value of the new situation is denoted *q*. If a *new* hidden unit must be added, then the situation component of its connections with the input will be initialized using the situation vector *s*. The Q-value component will be initialized with the received

reinforcement *r*. The action weight will be initialized with the value of *a*. Figure 7 describes how to compute the expected maximum Q-value for the new situation.
For all experiments discussed in this section, the constants for the neural learning are η_a =0.2, η_s =0.01, η_Q =0.5, σ =0.4 and the *acceptance_thereshold*=0.1.

Input : situation *s* and Q-value *q*

Step 1. Choose the value *i** of the index *i* such that

$$i^* = \arg\max_i \left\{ y_i = e^{-\left((s-w_S)^t (s-w_S) + \left|(q-w_Q)/2\right|^2 \right)/\sigma^2} \right\}$$

Step 2. If y_{i^*} is greater than the a*cceptance_threshold*

$action = w_a(i^*)$

else

Add a hidden unit.

Figure 5. Using the RBF neural network as an associative memory. The factor ½ in the second term of the exponent of *e* is due to the Q-value range ([-1,1]). The sensor values are normalized between [0,1]. The values of the *acceptance_threshold* and σ^2 define the generalization degree of the network.

Input : situation *s*, action *a*,
Q-max of the new situation *q,* and the reinforcement *r*.

Step 1. Choose the value *i** of the index *i* such that

$$i^* = \arg\max_i \left\{ e^{-\left((s-w_S)^t (s-w_S) + |a-w_a|^2 \right)/\sigma^2} \right\}$$

Step 2. If $e^{-\left((s-w_S)^t (s-w_S) + |a-w_a|^2 \right)/\sigma^2}$ is greater than the *acceptance_threshold*

$w_Q(i^*) = w_Q(i^*) + \eta_Q \left(r + \gamma q - w_Q(i^*) \right)$

$w_S(i^*) = w_S(i^*) + \eta_S \left(s - w_S(i^*) \right)$

if *r* is positive

$w_a(i^*) = w_a(i^*) + \eta_a \left(a - w_a(i^*) \right)$

if *r* is negative

$w_a(i^*) = w_a(i^*) + \eta_a \left(1 - a - w_a(i^*) \right)$

else

Add hidden unit.

Figure 6. Weight update rule. η_a , η_s and η_Q are the learning rate constants for each kind of update ($0 < \eta_a , \eta_s , \eta_Q < 1$).

Input : situation *s.*

Step 1. Choose the value *i** of the index *i* such that

$$i^* = \arg\max_i \left\{ y_i = e^{-\left((s-w_S)^t (s-w_S) + \left|(1-w_Q)/2\right|^2 \right)/\sigma^2} \right\}$$

Step 2. Return as Q-max the value of $w_Q(i^*)$

Figure 7. Neural network use to obtain the Q-max value of a new situation.

4.2 Experiments with a robot arm.

The use of Reinforcement Learning in order to synthesize behaviors for robot arms has antecedents in the works of Tham [26,27] or Martín and Millán [13,14]. This type of robot presents specific difficulties compared to the more conventional mobile robot:

1. The area of work is not only fixed by the environment but also by their own mechanical limitations.
2. In general, the system of sensors is not prepared to perceive the environment, but its own internal state (sensors in their joints).
3. The kinematics and spatial form imply severe restrictions in the selection of the actions to be taken. For example, the action corresponding to avoid an obstacle must not only consider the location of the end effector but also the configuration of the arm, that is, the location of all the links in space.

Taking into account the above considerations, we have carried out a series of experiments with the robot arm Escorbot ERVII with the objective of studiyng two behaviors related with the recognition of the robot workspace. In the first case we consider the bound avoidance behavior in which the relationship # r+/it. (and #r-/it.) vs. value of the RF parameters is linear. In the second case, we also take into account the presence of obstacles. Thus, the proposed RFs use integral estimators for distance (first definition of the RF) and differential estimators for speed (second definition of the RF). Both definitions result in non-linear relationships #r+/it. and #r-/it. vs. value of the RF parameters and require the use of UPA. We also have carried out several experiments to show the use of UPA during learning (Dynamic UPA), however due to the size of this version we decided only to mention this issue for the mobile robot (see sections 4.4.1 and 4.4.2). The interested reader could read that in [21].

4.2.1 Workspace recognition by learning the bound avoidance behavior.

Unlike mobile robots, robot arms must possess an explicit knowledge of their workspace because they have spatial and mechanical restrictions Craig, [36], which impose limits on their actions even in obstacle-free environments. We consider the learning of the bound avoidance behavior as an appropriate form to obtain the workspace recognition. In this section, we present a series of experiments carried out with two consecutive links of the robot Scorbot ERVII (Fig. 8), both with rotational joints acting in the same plane. The joint sensors or encoders of the robot have a linear output function regarding the value of the perceived angles. We took advantage of this situation in order to evaluate the method of computing the RF-parameters proposed in the section 2.3.2.
Each situation is represented by a vector of two elements (s_1, s_2) where s_1 and s_2 are the encoder outputs of the joints 1 and 2 respectively. Each action is represented by a vector $(\Delta\theta_1, \Delta\theta_2)$, where $\Delta\theta_1$ and $\Delta\theta_2$ indicate the displacements to be produced in each joint during its execution.
The RF for the desired behavior is

$$RF(s_1,s_2)=\begin{cases} -1 & if \quad g_1(s_1)<\theta_- \quad or \quad g_2(s_1)<\theta_- \\ & or \quad g_3(s_2)<\theta_- \quad or \quad g_4(s_2)<\theta_- \\ +1 & if \quad g_1(s_1)>\theta_+ \quad or \quad g_2(s_1)>\theta_+ \\ & or \quad g_3(s_2)>\theta_+ \quad or \quad g_4(s_2)>\theta_+ \\ 0 & otherwise \end{cases}$$

where $g_1(x)=x-IB_1$, $g_2(x)=SB_1-x$, $g_3(x)=x-IB_2$ and $g_4(x)=SB_2-x$. IB_1, SB_1, IB_2 and SB_2 are the lower and upper bounds of angles for joint 1 and joint 2 respectively.

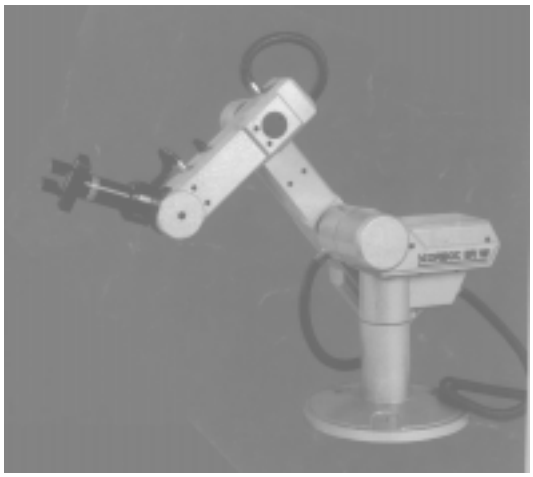

Figure 8. Scorbot ER VII. It has five axes, three for locating the end effector in the 3-dimensional workspace and two for orienting the tool on the end effector. Each axis has a DC motor driven by specific card in the control unit. The robot also has five incremental encoders in its joints. Each motor can be activated with an integer in the range from -5000 to 5000 traduced by the driver card from -24 volts to 24 volts. The encoders have a resolution of 0.004 º/count. The operating range of the axis are 250º, 170º, 225º, 180º and 360º from axes 1 to 5 (beginning from the base) respectively. The repeatability is 0.2 mm. The maximum workload is 2000 grams and the maximum speed is 1000 mm/sec.

Each experiment consists of a series of movements in the workspace of the manipulator. However, we only considered those movements carried out in a zone known as the Hot-Region: the set of situations, in which the arm approaches sufficiently close to its limits $HR(s_1, s_2) = \begin{cases} 1 & if \quad s_i \in [IB_i, IB_i + \alpha] \quad or \quad s_i \in [SB_i - \alpha, SB_i] \\ & \quad i = 1,2 \\ 0 & otherwise \end{cases}$

where α is a threshold value controlling the size of the Hot-Region.

The action vector is obtained by adding a random component (that decreases linearly during the whole experiment) to the action selected by the Policy. Each movement is the consequence of applying the action vector. In this way, the first movements will be characterized by a strong exploration component, and the last by a strong exploitation component.

The first step was to compute the value of the parameters θ_+ and θ_-, by considering balanced values for the ideal proportions of positive, negative and null rewards ($p_+ = p_- = p_0 = 1/3$). Since the situations are uniformly distributed, the probability that link 1 is in $(IB_1 \ , IB_1 - \theta_-)$ is $\frac{1}{2}\frac{(IB_1 - \theta_-) - IB_1}{\alpha}$ which can be written as $\frac{\theta_-}{2\alpha}$. Also, the probability that link 1 is in $(SB_1 - \theta_+, SB_1)$ is $\frac{1}{2}\left(1 - \frac{\theta_+}{\alpha}\right)$. Similar results are obtained for link 2. Considering that, if the angle of link 1 is close to IB_1, it is not possible that the same angle is close to SB_1; and the probabilities corresponding to link 1 and to link 2 are independent, we can compute $\frac{\#r_-}{\#r_+}$ as $\frac{\#r_-}{\#r_+} = \frac{\theta_-}{\alpha - \theta_+}$.

We want $\frac{\#r_-}{\#r_+}$ to approach $\frac{p_-}{p_+}$. That is, $\frac{\#r_-}{\#r_+}$ has to approach 1. So, we obtain the first equation: $\alpha = \theta_- + \theta_+$.

Since the sum of desired ratios for each type of reinforcement is equal to 1, we obtain the second equation: $\alpha - \frac{1}{3} = \theta_- - \theta_+$. Solving the system for $\alpha = 0.3$ gives $\theta_- = 0.1$ and $\theta_+ = 0.2$.

The second step was to use Q-Learning (TD (0)), with the computed values of θ_+ and θ_-, to learn the bound avoidance behavior. The lower and upper limits imposed for the joints 1 and 2 were IB_1= 0.05, SB_1= 1.85, IB_2=2.4 and SB_2=1.33 respectively (all values in radians). The values of the learning rate constants in the weights updating rule of the neural network were $\eta = 0.01$, $\eta_u = 0.3$, $\eta_a = 0.1$, $\sigma = 0.5$ and $\gamma = 0.5$.

Figure 9 shows three learning indicators. The first (box A) is the ratio #overbounds/it. It is a measure of the robot performance in carrying out actions only inside the workspace. The second indicator is the average of the distance (in radians) obtained after each movement. The positive slope of the curve shown in the box B indicates a correct learning evolution (i.e., each time distances are farther away from the limits). The third indicator (box C) is the sum of reinforcements and shows the suitability of the obtained behavior to the specified RF.

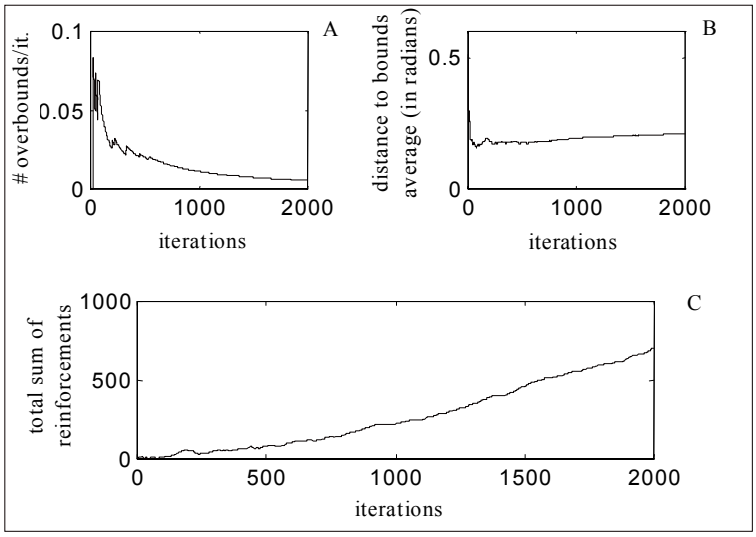

Figure 9. Indicators of the learning of the bound avoidance behavior. The RF-parameters were computed before the learning (θ_+=0.2, θ_-=0.1).

We may suppose that an appropriate choice of the parameters is a key factor in obtaining the desired behavior. Therefore, we carried out 5 experiments with different values of θ_+ and θ_-. The results (see [21]) showed that a balanced proportion of each kind of reinforcement guarranties the best performance in terms of the expectation of rewards. Then, we can conclude that the computation of the RF parameters for the linear case, with a balanced ratio of reinforcements, allows an appropriate sampling of the situation-action space resulting in an effective clustering.

4.2.2 Workspace recognition by the learning of bound and obstacle avoidance behavior

In this section we consider the learning of bound and obstacle avoidance behavior for the workspace recognition characterized by the presence of obstacles. The perception of the environment implies the robot arm has proximity sensors along it. We use a set of virtual sensors. They follow a distribution similar to the proposal made by [Martín and Millán, [14]. Figure 10 schematizes the location of the 13 virtual sensors. The transfer function of each one is defined as for standard infra-red sensor. Their output are given by

$$s_i = \sum_{j=1}^{N_{obst}} \left(1 - d_{ij} \big/ d_{scope}\right) e^{-\left(\frac{ang_{ij}}{ang_{scope}}\right)^2}$$, where $i = 1..13$, N_{obst} is the number of obstacles, d_{ij} and ang_{ij} represent the distance and angle between the sensor i and the obstacle j. d_{ij} and ang_{ij} are considered only if the obstacle j is inside the operative field of the sensor i. The parameters d_{scope} and ang_{scope} define the operative field of the sensors. There are 4 additional virtual sensors to detect bounds in the workspace of the simulated arm. The output of these sensors is defined by

$$s_{14} = (IB_1 + \alpha - \alpha_1)/\alpha,\ s_{16} = (IB_2 + \alpha - \alpha_2)/\alpha,$$
$$s_{15} = (\alpha_1 - SB_1 + \alpha)/\alpha \text{ and } s_{17} = (\alpha_2 - SB_2 + \alpha)/\alpha,$$

where IB_1, SB_1, IB_2 and SB_2 are the lower and upper bounds of angles for joints 1 and 2 respectively, α_1 and α_2 are the angles sensed by the encoders of the joints 1 and 2, respectively, and α is a threshold value. This type of situation vector representation allows, after learning, the use of the obtained behavior in environments different from that used during learning.

Again, we have used 2 consecutive links of the Scorbot ER VII, both with rotational joints acting in the parallel planes. While the representation of a situation is given by a description of the 17 virtual sensor values, the representation of the action is given by the vector angle changes in the joints $(\Delta\theta_1, \Delta\theta_2)$.

We have performed a series of experiments. Each experiment consists of a series of trials. Each trial is a sequence of 100 movements starting from a random initial position. Each movement is the result of applying the vector $(\Delta\theta_1, \Delta\theta_2)$ to the joints of the manipulator. This vector is obtained from the current Policy plus a random component (exploration) decreasing linearly as the whole experiment continues. Learning and UPA algorithm are activated in the Hot Region which is defined by

$$HR(s_1,..,s_{17}) = \begin{cases} 1 & if \quad \left(\sum_{i=1}^{13} s_i + \sum_{i=14}^{17} 4 s_i\right) > 0.5 \\ 0 & otherwise \end{cases}.$$

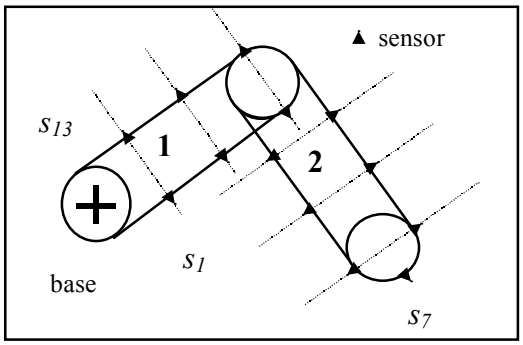

Figure 10. Distribution of the virtual sensors (filled triangules) along the arm. Links 1 and 2 are in parallel planes.

4.2.2.1 Using UPA to tune Reinforcement Functions based on integral estimators

As we mentioned in section 2.2.1, if the performance of the desired behavior can be expressed in terms of constraints on the distance (sensed by the robot), then the integral estimators can be used as an approximation of such distance in the definition of the RF. Based on this, we propose to study the following definition of the RF for the bound and obstacle avoidance behavior:

$$RF(s_1,..,s_{17})=\begin{cases}+1 & if \quad g_1(s_1,..,s_{17})<\theta_+ \quad is \quad true \\ -1 & if \quad g_1(s_1,..,s_{17})>\theta_- \quad is \quad true \\ 0 & otherwise \end{cases},$$

where $g_1\left(s_1,..,s_{17}\right)=\sum_{i=1}^{17}s_i$.

Before starting the learning, we have used UPA in order to tune the RF-parameters. The values of the desired ratios of positive and negative reinforcements (p_+ and p_-) were 0.35 for both cases. A complete analysis of the evolution of the values of θ_+ and θ_- during the execution of UPA can be found in [21].

We carried out the learning using the values obtained in advance by UPA (θ_+ =0.56, θ_- =0.76). Each experiment consisted of 20 trials. The left top of the figure 11 shows the evolution of the indicator "amount of collisions over iterations". A collision occurs when the robot bumps into an obstacle or exceeds some of the imposed limits to the angles of its joints. The right top of the same figure shows the evolution of the "average of the integral estimator of the distance" between the robot and their limits or obstacles (the less this value, the larger the distance). The bottom of the figure 11 shows the evolution of the "sum of reinforcements" during learning.

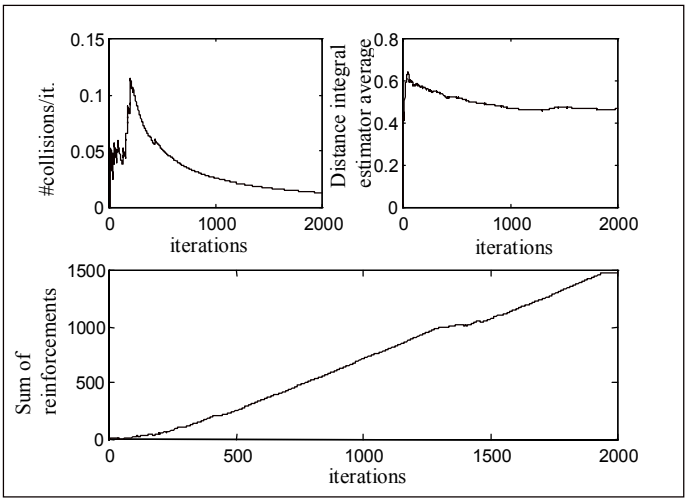

Figure 11. Learning indicators of the bound and obstacle avoidance behavior. The RF parameters values are obtained in advance using UPA (θ_+ =0.56, θ_- =0.76).

The values of desired ratios of penalties and rewards, p_+ and p_-, for the execution of UPA (before learning), try to guarantee an appropriate proportion of reinforcements during the initial phase of the learning (exploration). However, these values are just tentative, and it is possible to find other pairs that lead to appropriate learning. This experience (plots not showed in this compact version of the thesis) allows us to confirm that the (0.35,0.35.0.3) election was adequate.

4.2.2.2 Using UPA for tuning parameters in a RF based on differential estimators

To complete the RF expressivity, we defined a second RF based on differential estimator of the distance to obstacles:

$$RF\left(s_1^t,..,s_{17}^t,s_1^{t-1},..,s_{17}^{t-1}\right)=\begin{cases}+1 & if \quad g_1\left(s_1^t,..,s_{17}^t,s_1^{t-1},..,s_{17}^{t-1}\right)<\theta_+ \quad is \quad true \\ -1 & if \quad g_2\left(s_1^t,..,s_{17}^t\right)>\theta_- \quad is \quad true \\ 0 & otherwise \end{cases},$$

where $g_1\left(s_1^t,..,s_{17}^t,s_1^{t-1},..,s_{17}^{t-1}\right)=\sum_{i=1}^{17}\left(s_i^t-s_i^{t-1}\right)$ and $g_2\left(s_1^t,..,s_{17}^t\right)=\sum_{i=1}^{17}s_i^t$.

First, we have executed UPA with values of desired ratios $p_+=p_-=0.35$, and have obtained the values of 0.04

and 0.71 for θ_+ and θ_- respectively. Then we conducted an experience of 20 trial putting the found values into the RF. Figure 12 shows the evolution of the "amount of collision over iterations", the "average of the integral estimator of the distance" and the "sum of reinforcement".

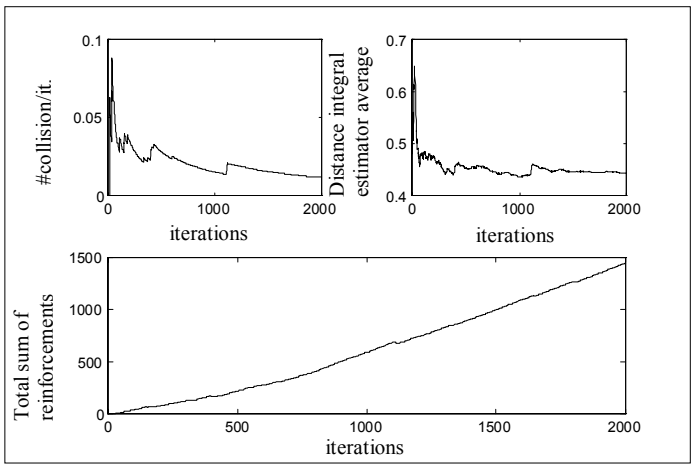

Figure 12. Learning indicators of the bound and obstacle avoidance behavior using differential estimators in the RF. The values of the parameters were obtained in advance using UPA (θ_+=0.04, θ_-=0.71).

4.3 Conclusions about the behavior learnings with a robot arm

We have studied two robot arm behaviors. In the first case – bound avoidance - we evaluated the useful of the RF parameter computation when the relations between reinforcements and parameters are linear. Under these circunstances, our hypothesis about the importance of a balanced ratio of reinforcements during the exploration phase was confirmed. In the second case – bound and obstacle avoidance - we evaluated the use of UPA for tuning the parameters for reinforcement functions which used two different kind of estimators: integral and differential ones. Again, even under different circunstances, our hypothesis was confirmed.
Differential estimator and integral estimator gave similar results in terms of the performance. In this sense, the objective of selecting each one was not demonstrate what better is, but how their use can carry out.
Figure 13 illustrates the result of several actions during the greedy use of the obtained behavior in an environment different from that used during learning. The location was changed as well as the sizes of the obstacles. Collision during the 1000 movements of the experiment was not detected. The behavior observed corresponding to the bound and obstacle avoidance sinthesized with differential estimator (described in previous section).

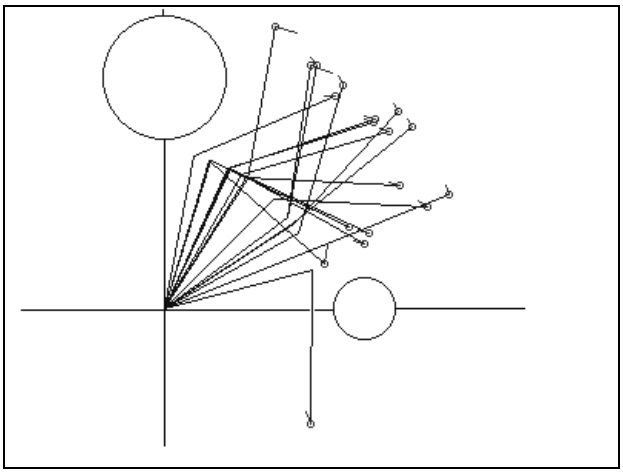

Figure 13. Representation of the actions associated with diverse units of the neural network obtained after learning with Dynamic-UPA of the bound and obstacle avoidance behavior. Differential estimators were used for the RF. The obstacles were changed (location and size). The values of the parameters were obtained in advance using UPA (θ_+=0.04, θ_-=0.71). The longer segments represent the links of the manipulator. The smallest segments on the end effector represent the speed vector in Cartesian space (the circle indicates the beginning). The speed was obtained by transformation of the associated actions using the kinematics jacobian.

4.4 Experiments with a mobile robot

We have used the Khepera robot (Fig. 14) in experiments for synthesizing obstacle avoidance and wall following behaviors using its 8 infrared (IR) sensors. The typical arena is shown on figure 15. The Khepera robot [18] is represented inside the arena as a circle. The robot has 8 sensors non-uniformly distributed around its body. Khepera is a miniature robot (55 mm in diameter and 35 mm in height). It can receive commands via an RS232 channel (up to 38400 bps) or work in an autonomous way controlled by programs which can be loaded into its

memory and executed by a 32 bits CPU with a 16 MHz microcontroller. Its sensing system comprises: 1) eight infra-red transmitter/receiver pairs along its perimeter which can detect obstacles up to a distance of 50 mm (10 bits resolution); and 2) two encoders associated with the wheels (a precision of 12 counts per millimeter). The speed of their motors can be set in the integer range [-10,10].

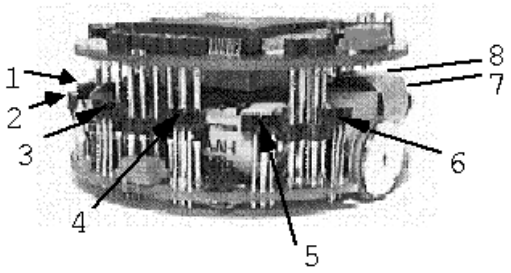

Figure 14. The Khepera mobile robot (Khepera is a product of K-Team Company, Switzerland). Sensors 1 - 6 are in front, 7 - 8 are in rear.

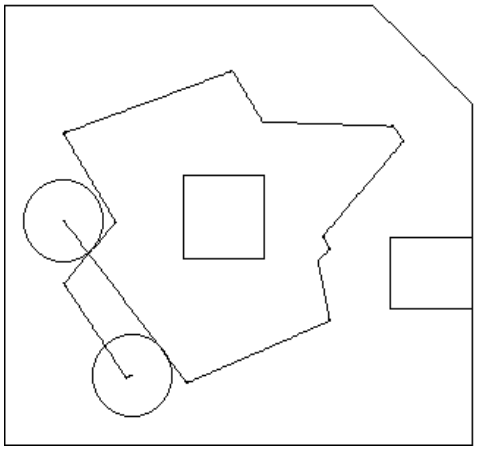

Figure 15. One of the arenas used in our experiments. Trajectories followed by Khepera (circle) during the obstacle avoidance test (after learning using Dynamic-UPA).

With the aim of reducing the problem dimensionality, we have chosen to represent the actions of the robot by scalar number representing the robot deviation angle (instead of using a vector of wheel speed). In this way, each action is followed by a forward movement in the selected direction. This type of representation is used in robots such as the Nomad 200. Detail of how it is used to conduct similar experiments to those described in this section can be found in [21].

4.4.1 Obstacle avoidance

The obstacle avoidance behavior is characterized by the robot reactions when one or more obstacles appear in its surrounding. A typical expression of what one could hope of this behavior is represented, for example, by the Braitenberg strategy [4]. We have used a RF definition similar to those used by [22] and [32]. In order to carry out the learning, we have endowed the robot with two reflexes:

Reflex 1: *if there is no sensor signal then move forward*, and
Reflex 2: *if a collision is detected then undo the last forward step*. [2]

Typically, the proximity sensor gives an estimate of the distance between the robot and an object, and therefore, it could be used as a measure of the robot relative position in the surrounding space. Since the sensors in Khepera measure the area free of objects around it, the integration of the perceived area of each sensor gives a measure of the average distance of the robot with respect to close objects. Thus, the relative robot position *estimator* is: $\tilde{p} = \sum_{i=1}^{6} s_i + 2\sum_{i=7}^{8} s_i$, where s_i is the sensor output i and the factor 2 in the second term is used in order to compensate for the non-homogeneous distribution of the sensors along the perimeter. So, the RF definition is

$$RF\left(s^t, s^{t-1}\right) = \begin{cases} +1 & if \quad g_1\left(s^t, s^{t-1}\right) < \theta_+ \\ -1 & if \quad g_2\left(s^t\right) > \theta_- \\ 0 & otherwhise \end{cases},$$

[2] The Reflex 2 will only be active during learning and it will be removed during testing.

where $g_1(s^t, s^{t-1}) = \sum_{i=1}^{6}(s_i^t - s_i^{t-1}) + 2\sum_{i=7}^{8}(s_i^t - s_i^{t-1})$ and $g_2(s^t) = \sum_{i=1}^{6} s_i^t + 2\sum_{i=7}^{8} s_i^t$.

We used UPA before learning to obtain the parameters for $p_+ = 0.3$ and $p_- = 0.3$. With the values obtained (θ_+ =0.2, θ_- =1.80), two groups of experiments were carried out (5 using only Q-Learning and 5 using Q-Learning with Dynamic-UPA). In the first experiment the learning was prolonged for 500 iterations, and in the remaining 9, only for 200 iterations (enough to synthesize an appropriate behavior). During Dynamic-UPA, p_{+ideal} and p_{-ideal} were set to 0.5 and 0.1, respectively. Figure 16 shows the results of two representative experiments of each group. The left top part shows the evolution of the index "number of collisions over iterations". The right top part of the same figure shows the evolution of the "average of the position estimator" (the lower this value, the higher the distance from the robot to surrounding objects)[3]. Finally, the bottom part of the figure shows the evolution of the sum of the reinforcements during learning.

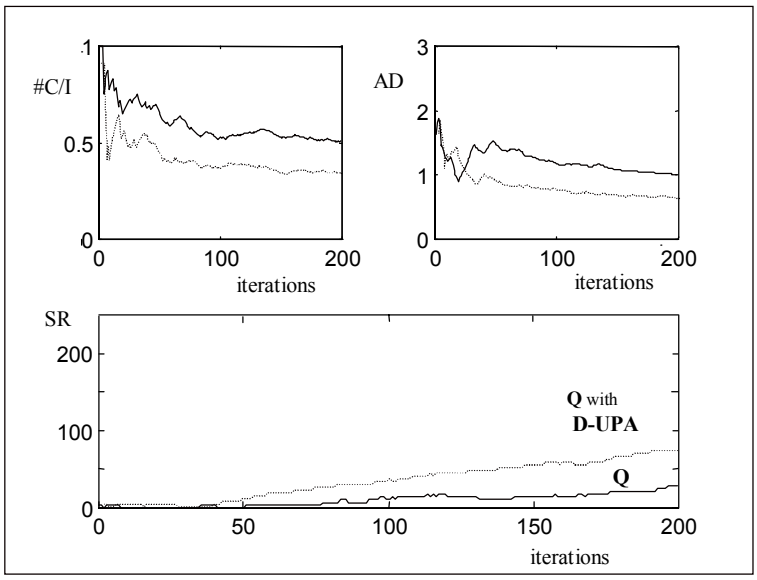

Figure 16. Evolution of the Q-Learning (with the parameter values computed previously by UPA) and Q-Learning with Dynamic-UPA (dashed line). #C/I is the number of collisions over iterations, AD is the average of the position estimator and SR is the sum of reinforcements (p_{+ideal} =0.5 and p_{-ideal} =0.1).

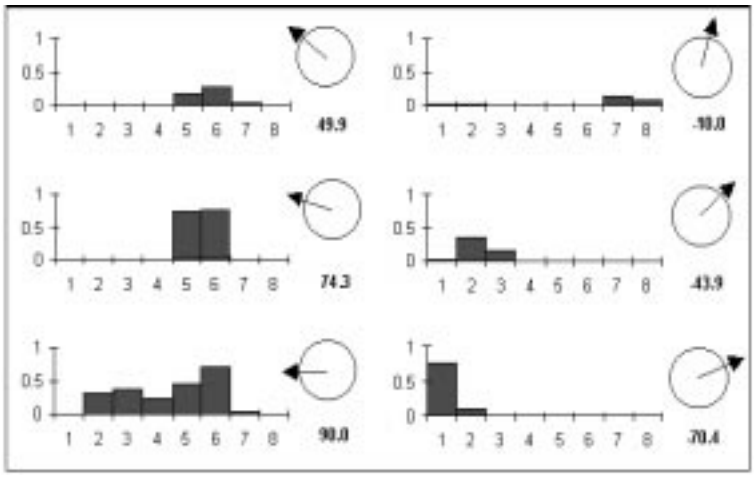

Figure 17. Representation of some of the situation-action pairs encoded by the RBF network (Dynamic-UPA). The numbers on the horizontal axis of each box indicate the component of the situation weight vector (one for each sensor). The vertical axis indicates the value each component. On the right of each box the direction deviation of Khepera is represented (the value of the angle is expressed in degrees).

Figure 17 shows the representation of some situation-action pairs corresponding to the connection weights of the RBF hidden units (clusters) used in the trajectories of figure 6. The network obtained has 41 hidden units, of which 29 have a positive associated Q-value and 11 of them a Q-value greater than 0.7. The hidden units with an associated negative value are due to that the network represents to the complete Q-function. Most of these units were added during the first part of the learning, and they are useful for updating of the w_Q (see Fig. 4) during the

[3] Each output sensor varies between 0 and 1023 (normalized usually between 0 and 1). The maximum output corresponds to closer objects. The minimum output corresponds to objects further away of 5 cm.

whole process. During the test of the synthesized behavior, only the hidden units with an associated positive Q-value make sense because the agent always tries to select the action associated with the current situation and the highest Q-value.

4.4.2 Wall following

The wall following behavior has been studied by several authors using different approaches. For example, Glorennec and Jouffe [6] use a set of fuzzy rules where the election of the consequent part is tuned by Q-Learning. A similar contribution is due to Godjevac and Steele [7] who propose a neuro-fuzzy controller based on RBF networks. Also, Donnart and Meyer [5], obtain a behavior for surround interposed objects toward the objective (in the context of a more complex task) using the MonaLysa structure: a hierarchical classifier system organized in modules (one of them is for reinforcing the "force" of each rule in the system, based on a measure of internal satisfaction of the agent). We have not been able to locate any report, in the literature, of the synthesis of this behavior using exclusively RL techniques.

We propose the following RF definition for learning a wall following behavior (on the right-hand side) using the Khepera robot: *the robot will be rewarded if it moves parallel to the wall,* and *the robot will be punished if it goes away from the zone where the right sensors can detect the wall.* The formal expression is

$$RF\left(s^t, s^{t-1}\right) = \begin{cases} +1 & \text{if} \quad g_1\left(s^t, s^{t-1}\right) < \theta_+ \\ -1 & \text{if} \quad g_2\left(s^t\right) < \theta_- \\ 0 & \text{otherwise} \end{cases}, \text{ where } g_1\left(s^t, s^{t-1}\right) = \left| s_6^t - s_6^{t-1} \right| \text{ and } g_2\left(s^t\right) = s_6^t .$$

In order to conduct learning, we have endowed the robot with two reflexes (they will only be active during learning and they will be removed during testing.):

Reflex 1: *if there is no sensor signal then rotate 20 degrees (clockwise) and move forward*, and

Reflex 2: *if a collision is detected then undo the last forward step*.

First, we used UPA to obtain the values of θ_+ and θ_- setting the proportions of desired positive and negative reinforcements to $p_+ = p_- = 0.2$. Figure 18 shows the behavior of the parameters, $\#r_+^{\Delta t}/\Delta t$ and $\#r_-^{\Delta t}/\Delta t$ during UPA execution.

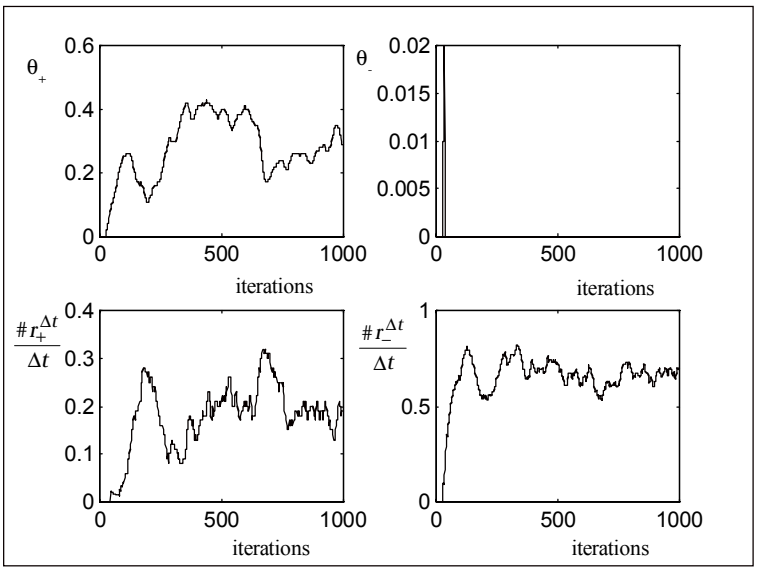

Figure 18. Evolution of the θ_+, θ_-, $\#r_+^{\Delta t}/\Delta t$ and $\#r_-^{\Delta t}/\Delta t$ during the execution of UPA (before learning) for tuning the RF corresponding to the wall following behavior ($\Delta\theta_+ = 0.02$, $\Delta\theta_- = 0.02$, $k_1 = k_2 = 0.9$ and $p_- = p_+ = 0.2$).

While an appropriate value was found for θ_+, one was not found for θ_- (the problem persisted when increasing the p_- value). The reason is that, during a purely exploratory phase, it is not very likely that the robot reaches positions with sensor 6 close to the wall. The plot of $\#r_-^{\Delta t}/\Delta t$ *vs. iterations* shows a particular situation: even as small as possible there is no value of θ_- which allows UPA (before learning) to obtain a proportion of negative reinforcements below ½. In this case, it is not possible to find values of the parameters that preserve a balanced ratio between negative and positive reinforcements with the guarantee of a minimum proportion of null reinforcements. A group of 10 learning experiments using Q-Learning were performed. Each one lasted 2000

iterations and the parameters values were θ_+=0.3 and θ_-=0.001 (θ_- could not be zero or the robot will never receive a negative reinforcement). In all cases, no acceptable[4] behavior was obtained. However, from these experiments we determined that $p_{+maximum}$=0.4. That is to say, the maximum feasible expected proportion of positive reinforcements using this technique of learning. We decided to use Q-Learning with Dynamic-UPA, initializing θ_+ at 0.3 and θ_- at 0.001. In 6 of the 10 experiments carried out we have obtained acceptable behaviors. Figure 19 shows the trajectory followed by the Khepera in a new environment. The strategy achieved by the Khepera to turn left (change counterclockwise to the wall direction) is different from the strategy to turn right. When the wall changes its direction clockwise, the Khepera can keep its distance with respect to the wall constant, producing a circular trajectory. In this case, small right turns are enough. However, when the wall changes its direction counterclockwise, the Khepera cannot keep its distance with respect to the wall and its trajectory cannot be parallel to it. In this case, large-enough left turns are needed. In this behavior, the largest left turn is (approximately) four times the largest right turn.

We have conducted a group of 5 additional experiments setting p_{+ideal} and p_{-ideal} at 0.3. All behaviors obtained are acceptable. We have recorded, for each iteration, the values of $\left|s_6^t - s_6^{t-1}\right|$, θ_+ and $\#r_+^{\Delta t}/\Delta t$. $\left|s_6^t - s_6^{t-1}\right|$ is an indicator of the quality of the actions performed (the actions must allow the Khepera to keep a constant distance with respect to the wall). θ_+ and $\#r_+^{\Delta t}/\Delta t$ are useful to observe the behavior of Dynamic-UPA. Figure 20 shows the evolution of these registers (corresponding to an experiment taken from this group). During the first 1500 iterations, Dynamic-UPA carried out a continuous process of expansion and contractions (of the positive and negative subsets). $\#r_+^{\Delta t}/\Delta t$ vs. iterations, shows many variations but the actions performed during exploitation do not match the desired Policy well (see plot of $\left|s_6^t - s_6^{t-1}\right|$ vs. iterations). Starting at iteration 1500, decrements of θ_+ (contractions) are accompanied by the exploitation of positively reinforced situation-action pairs which match the desired Policy well. That is, $\#r_+^{\Delta t}/\Delta t$ can maintain its value around p_{+ideal} while the difference $\left|s_6^t - s_6^{t-1}\right|$ shows a tendency to decrease. This fact occurs due to a systematic election of actions that join robot positions with, each time, more similar readings of sensor 6.

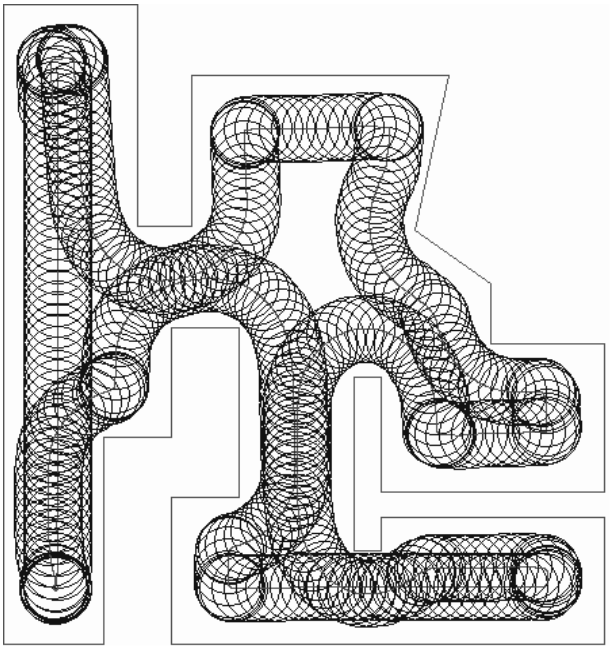

Figure 19. Testing of the synthesized wall following behavior. The environment is different from the one used during learning. The initial and final positions are located in the top left corner.

[4] A given behavior is acceptable when the Khepera follow a wall on the right-hand side holding the difference $\left|s_6^t - s_6^{t-1}\right|$ "close" or below θ_+. The robot should not bump against the wall nor should sensor 6 have a null output.

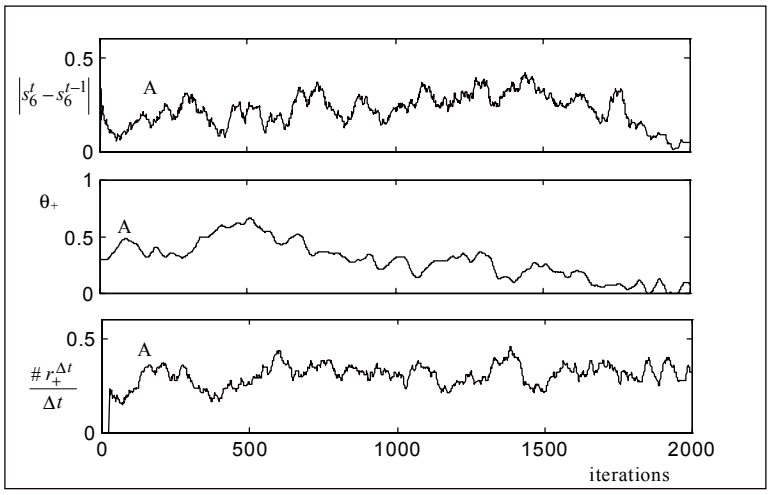

Figure 20. Evolution during learning of $\left|s_6^t - s_6^{t-1}\right|$, θ_+ and the performance measure ($\#r_-^{\Delta t}/\Delta t$).

Figure 21 shows a representation of the situation-action pairs corresponding to the connection weights of a few of the hidden units. The network has 113 units of which only 32 are associated with a positive Q-value (11 with a Q-value greater than 0.5). Boxes A to C represent actions that allow the Khepera to hold a parallel trajectory. The difference between the vector component value corresponding to sensor 6 in box A and B is hardly 10%, however it implies a change in the sign of the action (a new cluster had to be created by the network growing technique). In box C, the component corresponding to sensor 5 shows a marked increment with respect to the ones in box A and B. This means that the Khepera has lost its parallelism with respect to the wall and it must turn left to correct its direction. Boxes D and E correspond to situations where the Khepera is close to a dead-end hall (bottom corners of the figure 19). The values of the vector components in box D indicate the Khepera is closer to a wall direction change than in the case of the box E. So, the angle represented in box D is greater than the one represented in E. Box F represents a wall direction change counterclockwise. Boxes G and H represent corrective actions because the Khepera is, either very close or very far from the wall (these are, typically, used when the direction of the wall changes clockwise).

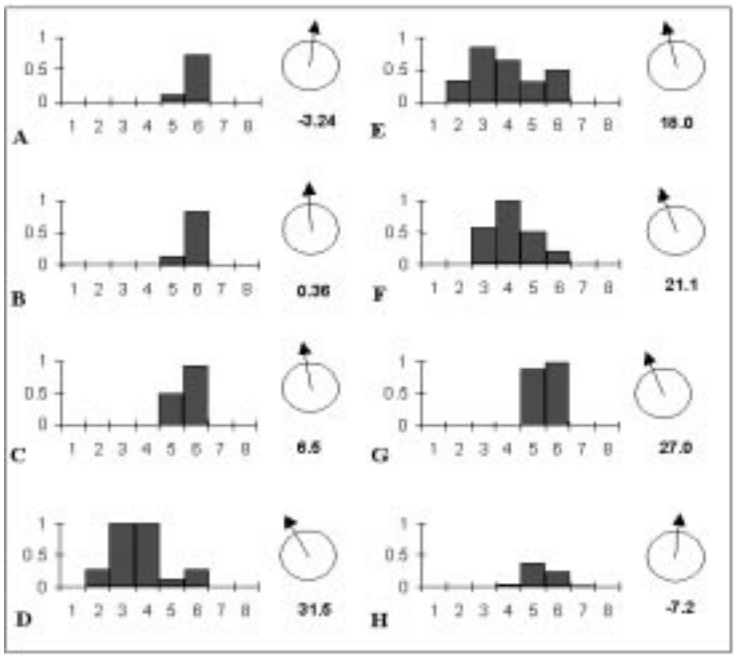

Figure 21. Representation of some of the situation-action clusters encoded by the network during Q-Learning with Dynamic-UPA (learning) of a wall following behavior. The numbers on the horizontal axis of each frame indicate the component of the weight vector. The vertical axis indicates the scale of the component values. On the right of each box the direction deviation of the Khepera is represented (the value of the angle is expressed in degrees).

6. Conclusion

In this dissertation, we have focused our research on the Reinforcement Function Design issue. We have differentiated two stages during the design. The first depends on the human capacity of translating the semantics of the task to be learned (usually specified in a natural language) into a set of simple constraints. We have separated the constraints according to the type of state variable estimator on which they act, in particular: position and velocity. The second requires the tuning of a set of parameter values, which allows us to obtain the final expression of the function. We have called this set the RF Parameters.

Our major effort has been the proposal of methods to obtain the values of these parameters. Using a particular, but representative RF expression, we have studied, during the exploration phase of learning, the relation between the Sum of each reinforcement type and the RF parameters. For linear relations, we have proposed an analytic method to obtain the RF parameters values (no experimentation required). For non-linear, but monotonic relations, we have proposed an algorithmic method: the Update Parameter Algorithm (UPA). We have shown that UPA can efficiently set the proportion of negative and positive reinforcements received during the exploratory phase of the learning. Finally, we have proposed using UPA during learning. Dynamic-UPA allows the whole learning process to maintain a desired ratio of positive and negative rewards. Thus, we introduce an approach to solve, for the first time, the exploration-exploitation dilemma - a necessary step for efficient RL.
We have carried out a series of experiments with real robots (a robot arm and miniature mobile robot) using RBF neural implementations with a growing technique. The results obtained during the experimentation confirm the utility of UPA to tune the RF parameters and also that Dynamic-UPA helps the synthesis of behaviors using Q-Learning. As an example, we have synthesized a wall following behavior with an original definition of the Reinforcement Function using Q-Learning and Dynamic-UPA.
Finally, there is not doubt about the convenience of having a technique to endow agents (and in particular robots) with Reinforcement Learning. Until today, there was no methodology allowing the designer to spend considerably less time and efforts to obtain the Reinforcement Function for a given problem. *For example, in one of the experiments described here, the Khepera mobile robot uses less than 45 minutes to obtain the wall following behavior, including the RF Parameters tuning – instead of weeks. The RF involved was built using only two constraints. We believe that this simple case is a good illustration of how RF design impacts RL applications.*

In our opinion, an appropriate RF design procedure should (1) define a set of constraints, (2) establish the logical relations among them, (3) tune (if necessary) the parameters associated with each constraint. We are confident that Dynamic-UPA allows optimally speeding up the learning (part 3), however convergence to the desired behavior is dependent on the expression of the RF itself (parts 1 and 2) - not only on the parameters values. Therefore, it is our desire to address these issues in the near future: What can the different elementary constraints be? What are the necessary constraints with respect to a given type of application? From this point of view, it is useful to characterize some elementary constraints according to the type of the state variable on which they operate (e.g., *position estimators* and *velocity estimators*). With respect to the part 3, certainly the choice of appropriate values for p_+ and p_- (or p_{+ideal} and p_{-ideal}) constitutes an aspect which must be studied. In fact, the motivation to use a balanced proportion of positive and negative reinforcements during UPA execution (to obtain a representative exploration) does not propagate to Dynamic-UPA. We overcame this difficulty by observing the expectation of these proportions (i.e. $p_{+maximum}$) after a simple Q-Learning experience. Currently, our aim is to find a rule for the automatic update of the desired proportion during the learning process, based on the expectation of the observers $\frac{\#r+}{t}$ and $\frac{\#r-}{t}$.

References

1. David Ackley and M. Littman, 1991. 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, 487-509.
2. Charles W. Anderson, 1993. Q-Learning with Hidden-Unit Restarting, in Advances in Neural Information Processing Systems 5, Morgan Kaufmann Publishers, 81-88
3. L.C. Baird, 1995. Residual Algorithms: Reinforcement Learning with Function Approximation, in Machine Learning: Proceedings of the Twelfth International Conference, Armand Prieditis & Stuart Russell, eds., Morgan Kaufman Publishers
4. Valentino Braitenberg, 1987. Vehicles. Experiments in Synthetic Psychology, Bradford Books, MIT Press (second printing).
5. Jean-Yves Donnart and Jean-Arcady Meyer, 1996. Learning Reactive and Planning Rules in a Motivationally Autonomous Animat*, IEEE Trans. on System, Man, and Cybernetics, Part B:Cybernetics*, **26(3)**, 381-395
6. Pierre Yves Glorennec and Lionel Jouffe, 1996. A reinforcement learning method for an autonomous robot, in Proc. of The 1st Online Workshop on Soft Computing, Org. By Nagoya University
7. Jelena Godjevac and Nigel Steele, 1998. Neuro-Fuzzy Control of a Mobile Robot, in Proc. of the Fourth International Conference on Neural Networks and their Applications, Marseille, France, 231-241
8. Mance E. Harmon, 1998. On line Reinforcement Learning Tutorial, University of Massachusets.(http://www~anw.cs.umass.edu/~mharmon/rltutorial/tut.html)
9. R. M. Kretchmar and C.W. Anderson, 1996. Comparision of CMACs and Radial Basis Functions for local Function Approximators in Reinforcement Learning, in Proc. of the International Conference on Neural Networks'97.

10. Long-Ji Lin, 1992. Self-Improving Reactive Agents Based on Reinforcement learning, Planning and Teaching, *Machine Learning*, **8**, 293-322.
11. Long-Ji Lin, 1993. Reinforcement Learning for Robots Using Neural Networks. PhD. thesis CMU-CS-93-103, Carnegie-Mellon University.
12. S. Mahadevan and J. Connell, 1992. Automatic Programming of Behavior-based Robots using Reinforcement Learning, *Artificial Intelligence* **55** 311-365.
13. Pedro Martín and J. del R. Millán, 1996. Learning Goal-Oriented Obstacle-Avoiding Strategies through Reinforcement for a Two-Link Sensor-Based Manipulator, *T. N. No.I.96.138, Joint Research Centre*, Ispra, Italy
14. Pedro Martín and J. del R. Millán, 1997. Combining Reinforcement Learning and Differential Inverse Kinematics for Collision-Free Motion of Multilink Manipulators, *LNCS 1240*, Springer Verlag, 1324-1333.
15. Maja J. Mataric, 1994a. Reward Functions for Accelerated Learning, in Machine Learning: Proceedings of the Eleventh International Conference, William W. Cohen and Haym Hirsh, eds. Morgan Kaufmann Publishers, 181-189.
16. Maja J. Mataric, 1994b. Learning to Behave Socially, in Proc. of Third International Conference on Simulation of Adaptive Behavior (SAB-94): From Animals to Animats *3*, D. Cliff, P. Husbands, J-A. Meyer and S. Wilson, eds., MIT Press, 453-462.
17. J. del R. Millán, 1996. Rapid, Safe and Incremental Learning of Navigation Strategies, *Special Issue on Learning Autonomous Robots, M. Dorigo Guest Editor, IEEE Trans. on Systems, Man and Cybernetics - part B*, **26**, No. 3, 408-420.
18. Francesco Mondada, E. Franzi and P. Ienne, 1993. Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms, in Proc. of the Third International Symposium on Experimental Robotics, Kyoto, Japan.
19. John Moody and Christian J. Darken, 1989. Fast Learning in Networks of Locally-Tuned Processing Units, *Neural Computation,* **1**, 281-294.
20. Juan Miguel Santos and Claude Touzet, 1999. Exploration Tuned Reinforcement Function, Special Issue of Neurocomputing (Elsevier Journal), 28/1-3, pp 93-105..
21. Juan Miguel Santos, 1999. Contribution to the study and the design of reinforcement functions, Ph-D thesis, Universidad de Buenos Aires, Universite d'Aix Marseille-III.
22. Samira Sehad and Claude Touzet, 1994. Reinforcement Learning and Neural Reinforcement Learning, in *Proc. of the European Symposium on Artificial Neural Networks*, in Proc. of the European Symposium on Artificial Neural Networks, Brussels, Belgium, April, 135-140.
23. Jürgen Schmidhuber, Jieyu Zhao, Nicol N. Schraudolph, 1997. Reinforcement Learning with Self-Modifying Policies, in Learning to Learn, Ed. by S.Thrun and L. Pratt, Kluwer Publishers, 293-309.
24. Richard S. Sutton, 1988. Learning to Predict by the Methods of Temporal Differences, *Machine Learning,* **3**, 9-44
25. Gerald Tesauro, 1992. Practical Issues in Temporal Difference Learning, *Machine Learning*, **8**, 257-277.
26. Chen K. Tham and Richard W. Prager, 1994. A Modular Q-Learning architecture for manipulator task decomposition, in Proc. of 11th. Conf. On Machine Learning, 309-317
27. Chen K. Tham and Richard W. Prager, 1993. Reinforcement Learning Methods for Multi-Linked Manipulator Obstacle Avoidance and Control, in Proc. of IEEE Asia Pacific Workshop on Advances in Motion Control, Singapore.
28. Sebastian B. Thrun, 1992. Efficient Exploration In Reinforcement Learning, Technical Report CMU-CS-92-102 Carnegie-Mellon University.
29. Sebastian Thrun, 1996. Exploration in Active Learning, in: M. Arbib ed., Handbook of Brain and Neural Networks.
30. Claude Touzet, 1994. Extending Inmediate Reinforcement Learning on Neural Networks to Multiple Actions, in Proc. of the European Symposium on Artificial Neural Networks, Brussels, Belgium, April, 153-159.
31. Claude Touzet, 1997. Neural Reinforcement Learning for Behaviour Synthesis, *Robotics and Autonomous Systems 22, Special issue on Learning Robot: the New Wave*, N. Sharkey Guest Editor, 251-281.
32. Claude Touzet and S. Sen, 1997. Learning Agents, invited paper at the First Conf. on Autonomous Agents, Marina del Rey, CA, February.
33. Christopher John Cornish Hellaby Watkins, 1989. Learning from delayed rewards, Ph.D. Thesis, King's College, University Library Cambridge
34. Christopher John Cornish Hellaby Watkins and Peter Dayan, 1992. Technical Note. Q-Learning, *Machine Learning*, **8**, 279-292.
35. Ping Zhang and Stéphane Canu, 1995. Indirect Adaptive Exploration in Entropy based Reinforcement Learning, in Proc. of the International Conference on Artificial Neural Networks, 354-359.
36. John J Craig, 1989. Introduction to robotics, mechanics and control (Second ed.). Addison-Wesley Publishing.