

Automatic Tuning of the Reinforcement Function

Juan Miguel Santos (#) and Claude Touzet¹² (*)

(#)Universidad de Buenos Aires, Cdad. Universitaria Depto. Computación, 1428 Bs As, Argentina.
Email: jmsantos@zorzal.dc.uba.ar

(*) Center for Engineering Systems Advanced Research, 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 tuning 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, in this case 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 learning, adaptive robotics, autonomous robot, reinforcement function, 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 way of programming robots by reward and punishment without needing to specify how the task (i.e., behavior) is to be achieved [Kaelbling, 1996]. 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. In reinforcement learning, the behavior is synthesized by using as a unique source of information a scalar, the so-called reinforcement, which evaluates behavior actions: 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. It promotes learning of relevant associations. There is no separation between a

learning phase and a utilization (or test) phase. 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. Models of the environment or of the desired behaviors are not required.

There has been a lot of research related to the issues of convergence [Dayan, 1994]; generalization [Mahadevan, 1992]; exploration [Thrun, 1992], [Zhang, 1996]; and memorization [Lin, 1992]. A major milestone in the course of RF development has been the proposal of neural-based implementations, which reduced learning time and memory requirements to realistic values, allowing true RL applications [Sehad, 1994]. Therefore, the RF has become the critical component. Despite all the ongoing research, there have been few efforts (if any) to propose a methodology or define a robust set of heuristics for the design of RFs. Authors usually report the building of the RF as an emergent process involving lots of trials and errors.

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.

² Part of this research was conducted during a previous appointment at DIAM-IUSPIM, Univ. of Aix-Marseille III, France.

In section 2, we introduce a general definition for a RF. In section 3, we propose an algorithm to compute during the learning phase the parameter values. In section 4, simulations and experiments involving the synthesis of avoidance obstacles are conducted with a mobile robot (Nomad 200). Finally, we discuss and conclude the obtained results and future work.

2. A Definition of the RF

A RF should always imply positive, negative, and null reward [Touzet, 1997b]. 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 can 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. 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, \dots, s_u) is the output readings of the sensors, $g_1()$ and $g_2()$ are any kind of functions.

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

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 moves value varies between 0.0 and 0.7. If θ_+ is small, the robot receives lots of positive rewards, and then the sum of the rewards will be large. The extreme case is when θ_+ is 0. As we increase θ_+ , the total amount of positive rewards diminishes. It must not reach 1000; otherwise, the robot would never receive positive rewards. The same reasoning applies to θ_- .

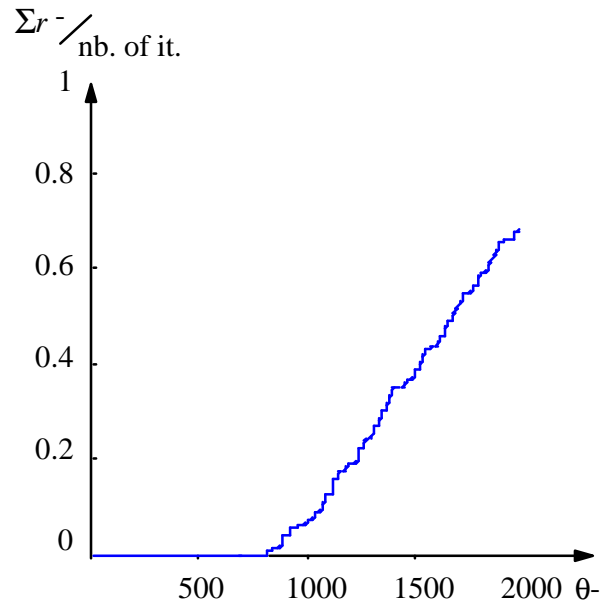
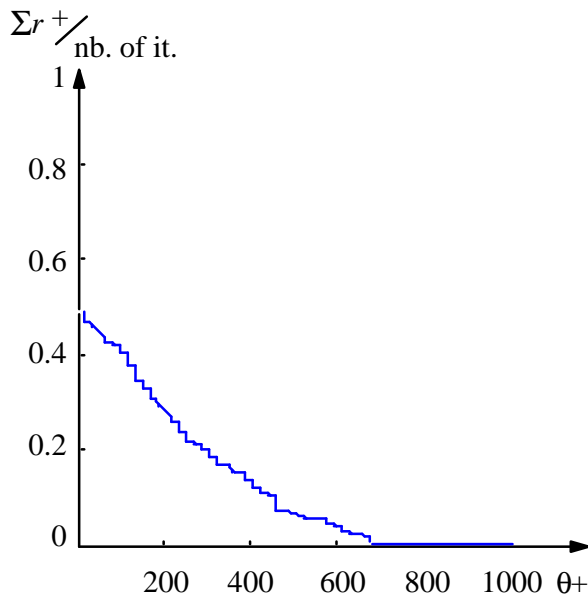


Figure 1. Each point of the graphs was obtained by performing 500 successive moves at random (using a random action generator for the mobile robot):

- a/ Sum of positive rewards over the number of moves versus the value of θ_+ [0 .. 1000].
- b/ Sum of negative rewards over the number of moves versus the value of θ_- [0 .. 2000].

3. Dynamic Estimation of Parameters in RF

We have derived an algorithm to dynamically compute during the exploration part of the learning phase the

parameter values. Let us assume that the functions $g_1()$ and $g_2()$, that associate the input situation to the RF value, are non-linear but monotonous. For the sake of

pedagogy, we assume here that $g_1(\cdot)$ is decreasing and $g_2(\cdot)$ is increasing (as shown on fig. 1). 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 behavior of (r_+ and r_- over time) should match. The problem is to adjust dynamically the threshold parameters of the positive and negative rewards (θ_+ and θ_-) so that we observe the convergences of the real observations (r_+ and r_- over time) towards the desired ratio (p_+ and p_-).

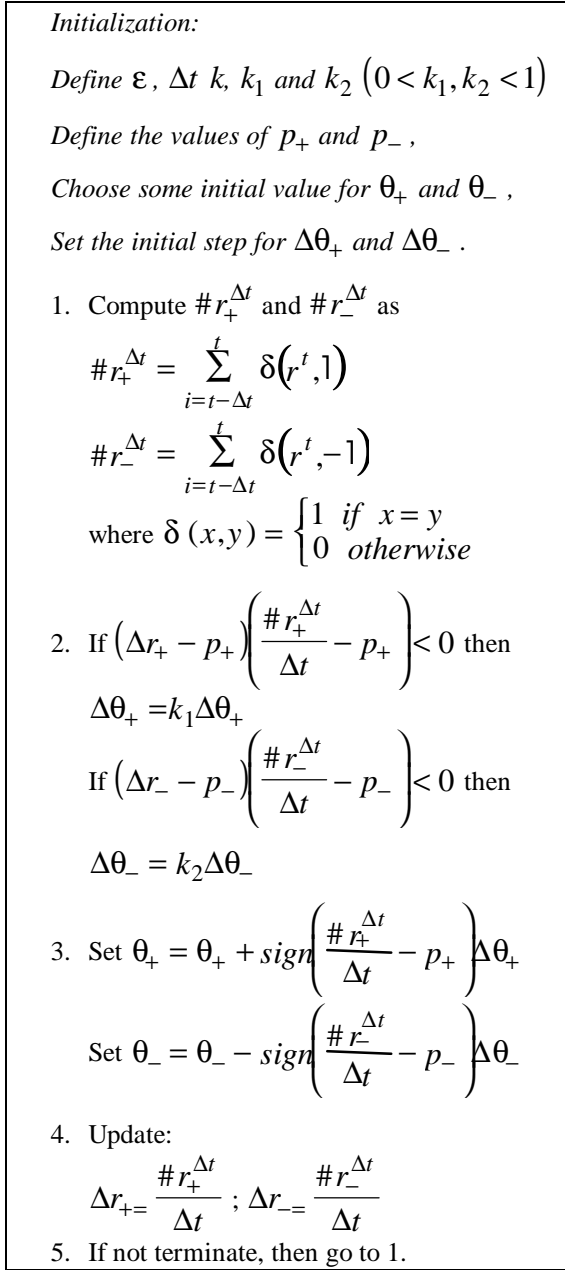


Figure 2. Update Parameters Algorithm.

The principle of the Update Parameters Algorithm (UPA) as described in figure 2 is to estimate the value of the real observations r_+ and r_- over time. To this end, two variables $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ are defined. They are the number of positive rewards occurring in the last Δt iterations ($\#r_+^{\Delta t} = \sum_{i=t-\Delta t}^t \delta(r^i, 1)$), where

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \text{ (resp. negative rewards}$$

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).

Each iteration, the values of $\#r_+^{\Delta t}$ and $\#r_-^{\Delta t}$ are updated. (Step 4 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_+ . Since the relation ($g_1(\cdot)$) between $\#r_+$ and θ_+ has been defined here (arbitrarily) as monotonously decreasing, an increment value ($\Delta\theta_+$) is added to θ_+ . In the opposite case ($\frac{\#r_+^{\Delta t}}{\Delta t}$ is smaller than p_+), then a decrement value ($\Delta\theta_+$) is subtracted to θ_+ . (First part of step 3).

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_- . Since the relation ($g_2(\cdot)$) between $\#r_-$ and θ_- has been defined here (arbitrarily) as monotonously increasing, a decrement value ($\Delta\theta_-$) is subtracted to θ_- . In the

opposite case ($\frac{\#r_-^{\Delta t}}{\Delta t}$ is smaller than p_-), then an increment value ($\Delta\theta_-$) is added to θ_- . (Second part of step 3 of UPA).

We may have crossed the optimal value for the threshold θ_+ (resp. θ_-) in which case the product

$$(\Delta r_+ - p_+) \left(\frac{\#r_+^{\Delta t}}{\Delta t} - p_+ \right) \text{ is negative (resp.}$$

$$(\Delta r_- - p_-) \left(\frac{\#r_-^{\Delta t}}{\Delta t} - p_- \right) \text{ is negative). Each time}$$

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:

$$\text{i) } \left| \frac{\#r_+^{\Delta t}}{\Delta t} - p_+ \right| < \epsilon \quad \text{and} \quad \left| \frac{\#r_-^{\Delta t}}{\Delta t} - p_- \right| < \epsilon$$

during k iterations, ($0 < \epsilon < 1$), or

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

4. 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.

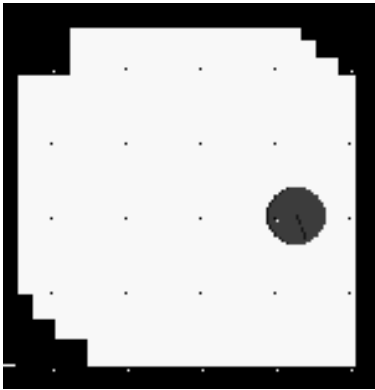


Figure 4. One of the arenas used in our experiments.

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.



Figure 3. The Nomad 200 mobile robot.

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 sensor values measure the distance to the obstacles (the greater the value, the farther the obstacles). 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 when the sum of the 8 front sensors is lesser than θ_- . This RF can be rewritten as:

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

where $g_1((s_1, \dots, s_{16})^t, (s_1, \dots, s_{16})^{t-1}) = (g_2(s_1, \dots, s_{16})^t + \sum_{i=7}^9 s_i^t) - (g_2(s_1, \dots, s_{16})^{t-1} + \sum_{i=7}^9 s_i^{t-1})$

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

we ignore some of the sensors (5,6,10,11,12) because they forbid the achievement of good learning performances.

Figure 5 shows the learning curve for θ_+ and θ_- during the exploration part when UPA is active. 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 r_+ and r_- to respect 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). The behavior of r_+ and r_- during the 500 UPA iterations is displayed on figure 6.

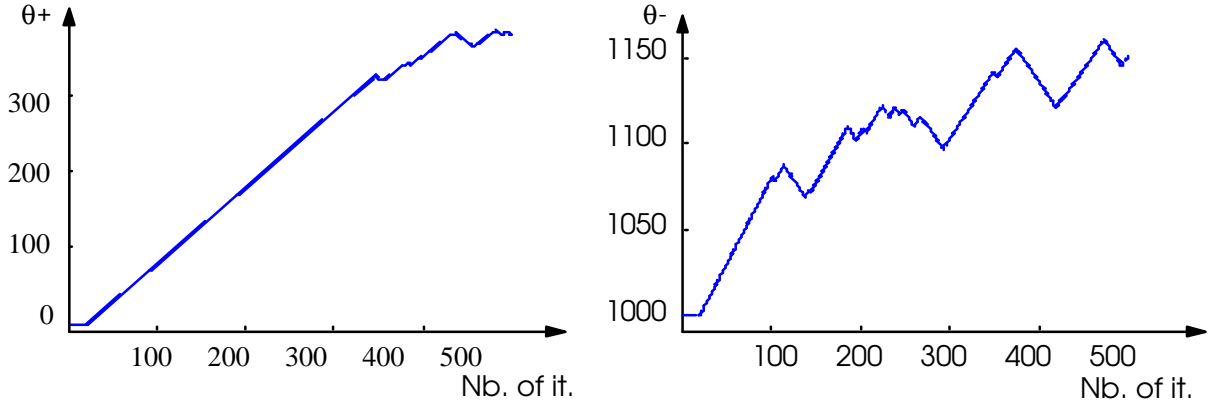


Figure 5. θ_+ and θ_- values during the 500 iterations of the UPA phase. The initial values are set randomly to 0.0 and 1000.0, respectively.

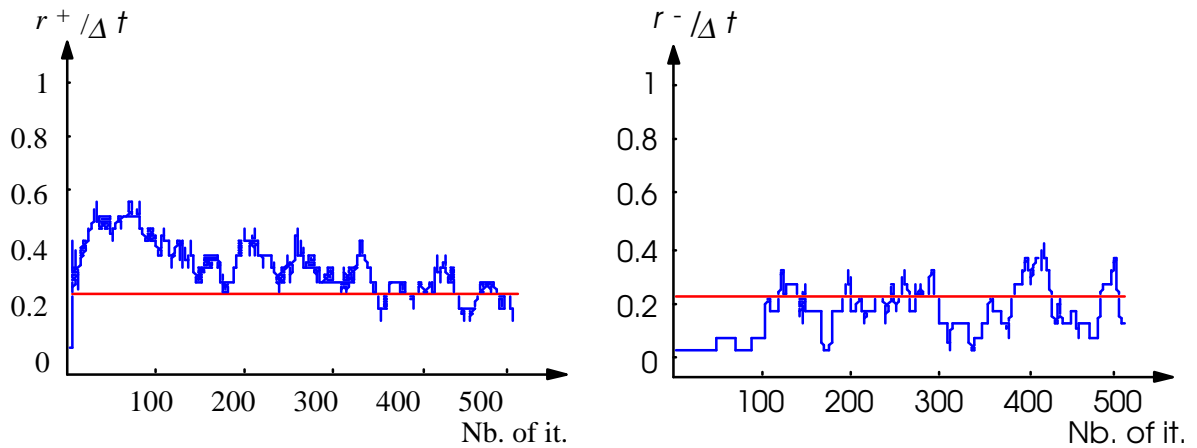


Figure 6. Values of $r_+/\Delta t$ and $r_-/\Delta t$ during the 500 UPA iterations. The oscillations are due to the limited size of the integration window (here $\Delta t = 20$ time steps) and the non-uniform distribution of the encountered situations. The objective is a ratio of 0.2 for r_+ and r_-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

After the 500 UPA iterations, Q-learning started the pure synthesis of the behavior using a self-organizing map implementation for memorizing and generalizing [Touzet, 1997a]. During the learning phase, the neurons of the self-organizing map approximate the probability density function of the inputs. The inputs are situation, action and the associated Q value (figure 7). The learning phase associates to each 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 to generalize across similar situation-action pairs.

The self-organizing map is used in the following way: the best action to undertake in a world 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). The selected neuron corresponds to a triplet (situation, action, Q value). It is this particular action that should offer the best reward in the world situation (fig. 8b).

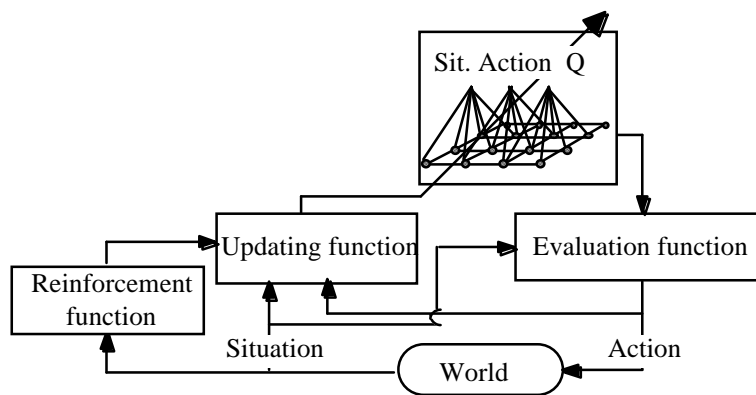


Figure 7. The self-organizing map implementation of the Q-learning.

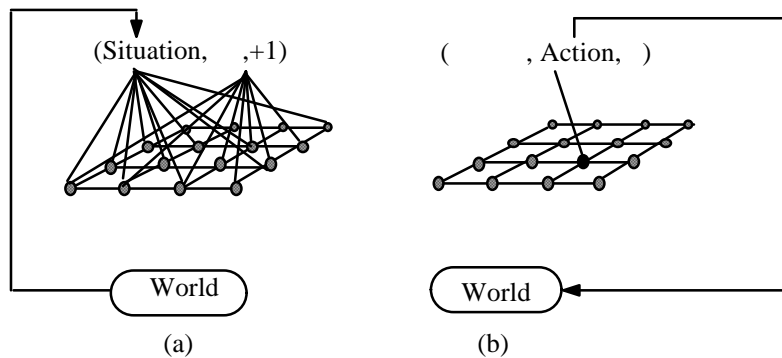


Figure 8. Selection of the best action to perform in the world situation.

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 selected neuron which weights give situation, Q value and the associated action.

The learning algorithm updates the Q value weight and, also, the situation and action weights. 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. 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 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 $r+/t$ and $r-/t$ during the experiments. During the initial 200 iterations, we can see the increase of $r+/t$ and the corresponding diminution of $r-/t$, representative of a good learning phase. The remaining 100 iterations are used to verify the level of

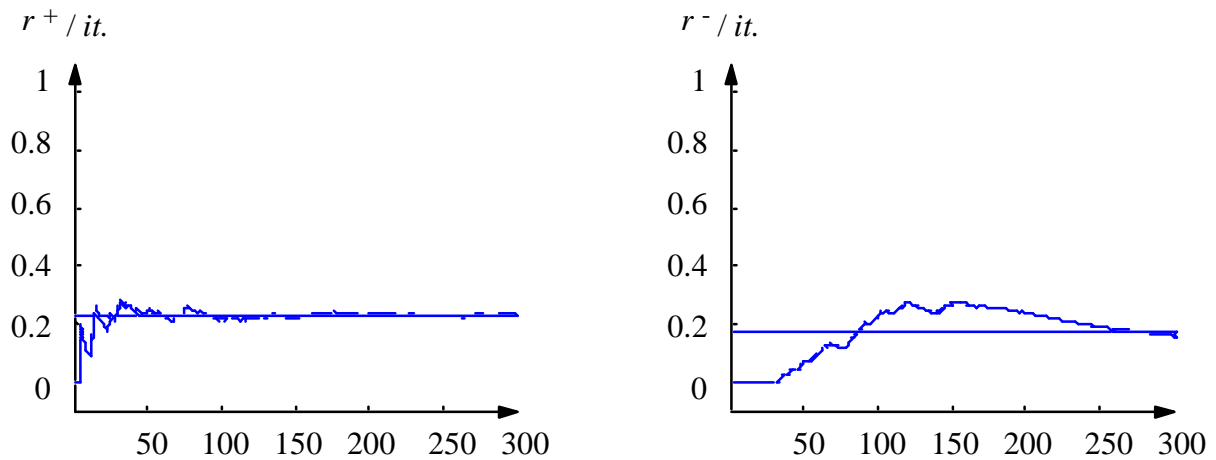


Figure 9. Graphs of $r+/t$ and $r-/t$ during the 200 learning + 100 test iterations.

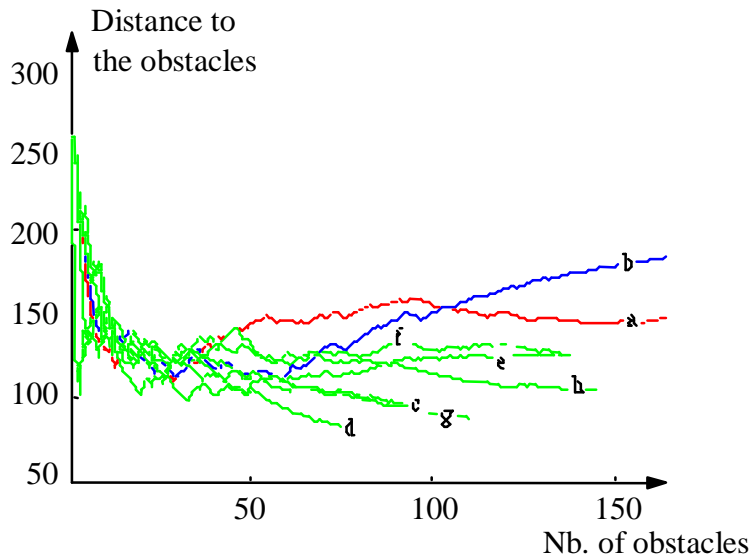


Figure 10. Distance to the obstacles vs. number of obstacles encountered during the 300 learning iterations.

- (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 ($\theta+ = 390$, $\theta- = 575$).

performance (test phase).

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 used RF parameters ($\theta+$ and $\theta-$) are accurate for this task. There are 200 learning iterations, after which there is no more learning for the last 100 iterations. We see that the learned behaviors avoid obstacles from a greater distance.

5. Conclusion

In this paper, we propose to solve in a particular case the so-called exploration/exploitation problem. Based on the assumption that the reinforcement function can be expressed has a constraint involving only one threshold parameter per case ($\theta_+ \rightarrow r_+$, $\theta_- \rightarrow r_-$), the proposed Update Parameter Algorithm (UPA) allows one to compute during a pure exploration part 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_+ and r_-) 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, suggesting that further experiments are certainly needed to really measure the impact of these first steps in RF design.

The UPA use has been restricted here to a pure (without learning) exploration phase. It would certainly be interesting to be able to use the UPA during the learning phase as well. However, the non-uniform distribution of the rewards during the learning phase imposes modifications of the algorithm previously described. It is our desire to address this issue in the near future.

Acknowledgment

The authors would like to thank L. Parker and the anonymous reviewers for helpful comments, suggestions and discussions.

References

- 1 Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, 4, pp. 237-285, 1996.
- 2 P. Dayan and T. Sejnowski, "TD(1) Convergences with Probability 1," *Machine Learning*, 14(3), 1994.
- 3 Sridhar Mahadevan and J. Connell, "Automatic Programming of Behavior-based Robots using Reinforcement Learning," *Artificial Intelligence* 55, pp. 311-365, 1992.
- 4 S.B. Thrun, "Efficient Exploration In Reinforcement Learning," *Technical Report CMU-CS-92-102*, Carnegie-Mellon University, 1992
- 5 Ping Zhang and Stéphane Canu, "Indirect Adaptive Exploration in Entropy-based Reinforcement Learning," *Proceedings of ICANN'96*, 1996.
- 6 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.
- 7 Samira Sehad and Claude Touzet, "Reinforcement Learning and Neural Reinforcement Learning," in *Proc. of ESANN*, Bruxelles, 1994.
- 8 Claude Touzet, "Neural Reinforcement Learning for Behaviour Synthesis," *Robotics and Autonomous Systems* 22, Special issue on Learning Robot: the New Wave, N. Sharkey Guest Editor, pp. 251-281, 1997a.
- 9 Claude Touzet and Sandip Sen, "Learning Agents," invited paper at the *First Conf. on Autonomous Agents*, Marina del Rey, CA, USA, 1997b.