UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Incorporating Fuzzy Logic to Reinforcement Learning

Gedson Faria
Roseli A.F. Romero

Nº 50

NOTAS

São Carlos - SP

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2577

Incorporating Fuzzy Logic to Reinforcement Learning

Gedson Faria
Roseli A.F. Romero

Nº 50

NOTAS

Série Computação

São Carlos – SP
Out./2000

Incorporating Fuzzy Logic to Reinforcement Learning

Gedson Faria and **Roseli A. F. Romero**
Departamento de Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo – Campus de São Carlos
Caixa Postal 668
13560–970 São Carlos, SP
gedson@icmc.sc.usp.br, rafrance@icmc.sc.usp.br

Resumo

Este artigo propõe um método de navegação baseado em sensores que utiliza lógica *fuzzy* juntamente com algoritmos de aprendizado por reforço, visando a navegação de robôs móvies em ambientes desconhecidos. É proposta uma codificação das leituras dos sonares como noções de distância através de conjuntos *fuzzy* e uma modificação no algoritmo R-learning através da incorporação de lógica *fuzzy*. Lógica *fuzzy* é utilizada para ponderar o reforço imediato, que é uma variável presente na maioria dos algoritmos de aprendizado por reforço. A confiabilidade do algoritmo modificado, R'-learning, foi verificada através de vários testes e comparações com o desempenho do algoritmo R-learning.

Abstract

This paper proposes a sensor-based navigation method that utilizes fuzzy logic into reinforcement learning algorithms for navigation of mobile robot in uncertain environment. The sonar readings are codified in distance notions by fuzzy sets and a modification in the R-learning algorithm by incorporating fuzzy logic is proposed. Fuzzy logic is used for weighting the immediate reward value, that is a variable presents in the most of the reinforcement learning algorithms. The effectiveness of the modified algorithm, R'-learning, is verified in several tests and compared to the performance of the R-learning algorithm.

1 Introduction

The problem of robot learning is essentially one of getting robots to do tasks without the need for explicitly programming them. Programming robots is extremely challenging, for many reasons. Sensors on a robot, such as sonar, behave in a complex unpredictable manner in typical unstructured environments, such as a crowded office building. Thus, understanding sensors is not sufficient; one has also model how they work in a particular task environment. To program a robot, the task has decomposed into a sequence of very low-level operations, such as moving joint angles or wheels. There are no good high-level robot programming languages that we can rely on. Thus, for these reasons there is considerable interest in robots that can automatically learn to do tasks.

Machine learning is a subfield of artificial intelligence (AI), whose ultimate goal is to replace explicit programming by *teaching*. Machine learning research has studied many different types of learning [1]. There are two types of learning: supervised and unsupervised. In the former, a teacher carefully selects examples for the learner, whereas in the latter the learner is given little or no feedback on the learning task.

Reinforcement Learning (RL) is an unsupervised learning paradigm and could be seen as a way of programming agents by reward and punishment without specify *how* the task is to be achieved.

Some difficulties in robot learning as: stochastic environment, real-time response and online learning are solved easily to reinforcement learning. Stochastic environment, or non-deterministic environment, is that in which to take the same action in the same state on two different occasions may result in different states and/or different reinforcement values. Robots must be capable of reactive planning, that is, a robot must respond to unforeseen circumstances in real time. For example, a robot operating in an office environment must be ever alert to the possibility of collisions from some unanticipated obstacle in its path, ranging from people moving randomly around to junk left around in corridors. Online learning is a desirable trait for any robot learning algorithm. The robot explores its environment to collect sufficient samples of the necessary experience.

Fuzzy logic combined with RL has been widely used for navigation of mobile robots and many works can be found in the literature. For example, Beom and Cho [2] proposed a sensor-based navigation method which utilizes fuzzy logic and reinforcement learning for navigation of mobile robots. In their work, fuzzy logic maps the input fuzzy sets, representing state space of mobile robot determined by sensor readings, to the output fuzzy sets representing the action space of mobile robot. A fuzzy rule base is built through the reinforcement learning. Yung and Ye [3] presented an alternative training approach to the EEM-based training method and a fuzzy reactive navigation architecture for an intelligent mobile vehicle navigator.

In reinforcement learning, the environment states must be mapped as discrete value set. Therefore, there is a loss of accuracy founded in the continuous value received of the real world. In this paper, the sonar readings are codified in distance notions by fuzzy sets improving the environment states set. Further, we are proposing a modification in the R-learning algorithm by incorporating fuzzy logic. This modification consists of weighting the reward value by the mean of membership degree of the fuzzy sets mapping the sonar readings. This paper is organized as follow. In Section II, a brief introduction to RL is presented including two known reinforcement algorithms, Q-learning and R-learning. The environment definition, the codification of the sonar readings by using fuzzy logic and a modification of the R-learning

algorithm are proposed in Section III. A comparative analysis between the R-learning algorithm and modified R-learning is performed in Section IV and finally future work is proposed in Section V.

2 Reinforcement Learning

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. RL is defined not by characterizing learning algorithms, but by characterizing a learning problem. Any algorithm that is well suited for solving that problem is considered a reinforcement learning algorithm. A full specification of the reinforcement learning problem in terms of optimal control is based on Markov decision processes. However, the basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal. Clearly such an agent must be able to sense the state of the environment to some extent and must be able to take actions that affect that state. The agent must also have a goal or goals relating to the state of the environment. In the standard RL model, an agent is connected to its environment via perception and action, as depicted in Fig. 1. On each step of interaction, the agent receives as input, i, some indication of the current state, s, of the environment; the agent then chooses an action, a, to generate an output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar *reinforcement signal*, r. The agent's behavior, B, should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a variety of algorithms. The agent's job is to find a policy π, mapping states to actions, that maximizes some long-run measure of reinforcement.

Formally, the model consists of a discrete set of environment states, S; a discrete set of agent actions, A; and a set of scalar reinforcement signals; typically $\{0,1\}$, or the real numbers.

The Fig. 1 also includes an input function, I, which determines how the agent sees the environment states. In this work, the fuzzy functions codify the sonar signals in indications of the current state.

A well-known form used to model RL problems is the Markov Decision Process (MDP). The remaining of this section is destined to present MDP and to detail the algorithms Q-learning and R-learning, which use MDP to reach an optimal policy.

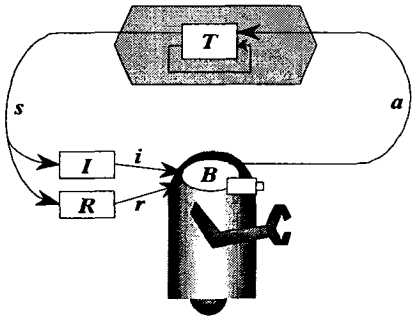

Figure 1: The standard reinforcement learning model.

2.1 Markov Decision Process

A state signal that succeeds in retaining all relevant information is said to be Markov, or to have the Markov property. For example, a checkers position—the current configuration of all the pieces on the board—would serve as a Markov state because it summarizes everything important about the complete sequence of positions that led to it. Much of the information about the sequence is lost, but all that really matters for the future of the game is retained. Similarly, the current position and velocity of a cannon ball is all that matters for its future flight. It doesn't matter how that position and velocity came about. This is sometimes also referred to as an "independence of path" property because all that matters is in the current state signal; its meaning is independent of the "path", or history, of signals that have led up to it.

Formally, a MDP consists of:

- a set of environment states, S
- a set of actions, $A(s)$
- a state transition probability function, $P(s, a, s')$
- a set of rewards to such state transition, $R(s, a, s')$

where $s, s' \in S$; $a \in A(s)$

There are many good references to MDP models [4], [5], [6], [7].

2.2 Reinforcement Learning Methods

The model consists of knowledge of the state transition probability function $P(s, a, s')$ and the reinforcement function $R(s, a, s')$. Reinforcement learning is primarily concerned with how to obtain the optimal policy when such a model is not known in advance. The agent must interact with its environment directly to obtain information which, by means of an appropriate algorithm, can be processed to produce an optimal policy.

The *model-free methods* learn a controller without learn a model. On the other hand, the *model-based methods* learn a model, and use it to derive a controller. More details about these methods can be found [8]. To follow, two model-free algorithms are presented: the Q-learning and R-learning because they are essential for the comprehension of the present work.

2.2.1 Q-learning

The Q-learning algorithm [9], [10] consists of updating the expected discounted reinforcement, $Q(s, a)$, by taking action a in state s. The Q-learning rule is

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], \tag{1}$$

where γ is the discount factor used to guarantee that the values are finite and α is the learning constant, and $0 < \alpha \leq 1$, $0 \leq \gamma < 1$.

After execute an action a, the agent leaves state s and it is going to state s', receiving by this action an instantaneous reward r. $max_{a'} Q(s', a')$ is the future evaluation of the reward for next state.

In addition, Q-learning is an exploration insensitive, which the Q values will converge to the optimal values, independent of how the agent behaves while the data is being collected

(as long as all state-action pairs are tried often enough). Q-learning is the most popular and seems to be the most effective model-free algorithm for learning from delayed reinforcement. However, it doesn't address any of the issues involved in generalizing over large state and/or action spaces. In addition, it may converge quite slowly to a good policy.

2.2.2 R-learning

Schwartz [11] examined the problem of adapting Q-learning to an average-reward framework. Although his R-learning algorithm seems to exhibit convergence problems for some MDPs, several researchers have found the average-reward criterion closer to the true problem than a discounted criterion and therefore prefer R-learning to Q-learning [12].

The algorithm R-learning has similar rule to Q-learning, being based on the deduction of values $R(s,a)$, and should choose actions a in a state s. For each situation, the learner chooses the action that has the largest value R, excepting sometimes in which it chooses an action any. The values of R are adjusted to each action, based on the following learning rule:

$$R(s,a) \leftarrow R(s,a) + \alpha[r - \rho + max_{a'}R(s',a') - R(s,a)], \tag{2}$$

that differs of the rule of Q-learning, simply for subtracting the medium reward ρ of the immediate reinforcement r and don't have discount γ for the next state, $max_{a'}R(s',a')$. The medium reward is calculated as:

$$\rho \leftarrow \rho + \beta[r - \rho + max_{a'}R(s',a') - max_{a}R(s,a)] \tag{3}$$

The key point is that ρ is always updated when a non randomly action was taken, $R(s,a)$. The medium reward ρ doesn't depend on some private state, it is a constant for the whole group of states. In the Fig. 2, the R-learning algorithm is presented. We have implemented R-learning algorithm adopting $\alpha = 0.2$ and $\beta = 0.001$ and the results obtained are presented in section 4. In spite the R-learning algorithm converges better than the Q-learning algorithm, the performance of both algorithms is sensible to choice of the free parameters α, γ and β. Another point is about the immediate reward value r. During the processing of both algorithms, the value is taken, exactly, as being the value of the reward assigned for each action and/or environment state. We are proposing in this work a new way of using the immediate reward value r. The new value, r', is taken as being a weighting value of the immediate reward r by using the fuzzy logic and will be described in next section.

Set ρ and $R(s,a)$, for all s, a arbitrarily
Repeat forever
  $s \leftarrow$ current state
  Choose action $a \in A(s)$
  Take action a, receive s' and r values
  $R(s,a) \leftarrow R(s,a) + \alpha[r - \rho + max_{a'}R(s',a') - R(s,a)]$
  if $R(s,a) = max_{a}R(s,a)$ then
    $\rho \leftarrow \rho + \beta[r - \rho + max_{a'}R(s',a') - max_{a}R(s,a)]$

Figure 2: R-learning Algorithm

3 Implementation

The proposed problem is to teach the robot Pioneer (Fig. 3) to navigate in a dynamic environment avoiding collisions. To solve this problem the R-learning algorithm was chosen because this algorithm, in general, has a model of behavior better than the Q-learning algorithm. To follow will be presented the robot used in our tests and a learning problem mapping proposed by us by incorporating fuzzy logic.

3.1 The Pioneer 1 Gripper robot

Pioneer 1 is a small and sophisticated mobile robot developed by Kurt Konolige of SRI International and Grinnell More of Real World Interface, Inc., and is available exclusively through *Activ*Media, Inc. Pioneer 1 has 7 ultrasonic sonar transducers to provide object

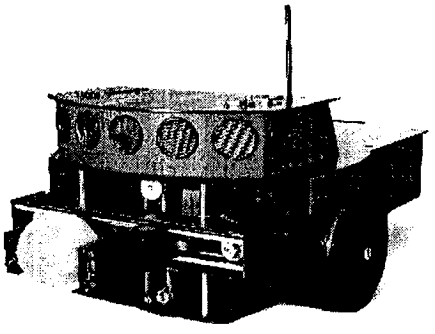

Figure 3: Pioneer 1 Gripper

detection, range information, and feature recognition. The sonar positions are fixed—one on each side and 5 forward facing—and attached to the inside of the console (Fig. 3). Sonar firing rate is 25Hz and sensitivity ranges from 20cm to over 5m.

The gripper can catch small objects and it has a vertical elevation system that allows catching objects with at least 2cm of height.

3.2 Learning Problem Mapping

The proposal of mapping of the navigation problem for the robot Pioneer 1 will be begun with the definition of the set of environment states proceeded by the set of actions and finally the set of rewards.

An environment state is considered, for the apprentice, as a distance notion of the objects. So, the problem of how to express this distance notion must be considered. One of the alternatives is just to divide the interval from 200 to 5000 in several subintervals, in which each sub-range is classified as a distance notion. A subdivision proposed for the Pioneer 1 robot sonars can be observed in Table 1. To reduce the number of states of the environment, the signals received by the seven sonars are classified as a distance to the left, a distance ahead and a distance to the right. The distance of objects ahead is adopted as being the smallest value received among the five front sonars. So, the five front sonar signals are transformed

Table 1: Classification of the sonar readings in distance notions

sonar readings	class
200 – 450	nearest
451 – 900	near
901 – 1675	far
1676 – 5000	farthest

as a single signal and therefore the robot receives, in true, only three signals as can be seen in Fig. 4.

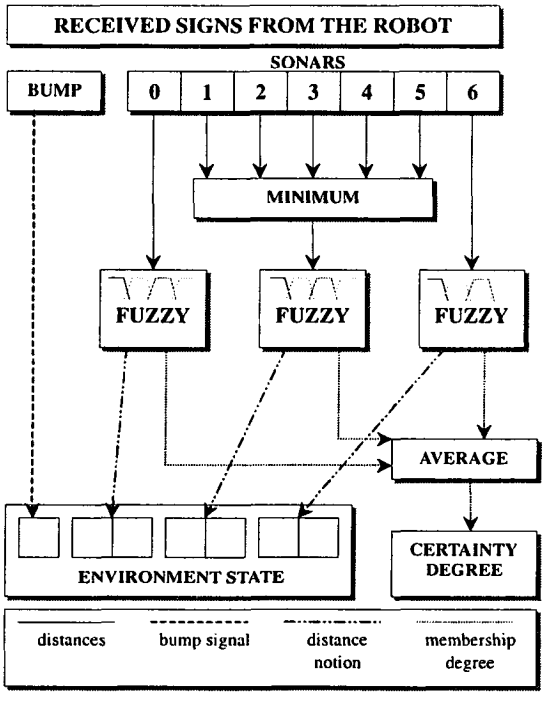

Figure 4: Definition of environment state.

The next step is to transform these three signs in distance notions, because the concatenation of these three values will represent an environment state. Table 1 could be used for this conversion process, but it treats equally all the values belonging to the same class, not differentiating those distance values closest to the court value of each sub-range. For example, according Table 1, the value 450 is classified as **nearest**, but this value is very close of the **near** class. For solving this problem, the use of fuzzy logic is proposed, so that the closest values to the value of court of each sub-range are not considered as precise as the others are. Four fuzzy sets are defined to represent the following linguistic concepts, **nearest, near, far, farthest**, and their membership degrees are presented in the Fig. 5. A new variable, *certainty degree*, denoted by φ, is also defined as being arithmetic mean of the membership degrees received from those fuzzy functions that codify the input signals (see Fig. 4). This variable will indicate the certainty degree about the state where the robot is. This variable is being proposed here due to the necessity to consider the fact that the robot can belong to the frontier

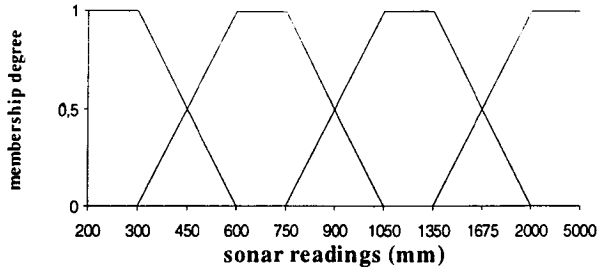

Figure 5: Four membership functions representing, respectively, the concepts: nearest, near, far, farthest.

of the two neighboring fuzzy sets. In this way, the reward value, r', will be given by:

$$r' = \begin{cases} r & \text{if the robot has certainty where it is} \\ \varphi r & \text{otherwise} \end{cases} \quad (4)$$

where φ is the mean of the membership degrees corresponding to the 3 fuzzy functions and r is the immediate reward.

It can be observed in Fig. 4 that the environment state is built through the concatenation of the distance notions obtained from 4 membership functions. Two bits were taken to represent the four distance notions. The state that indicates collision is represented with the bit of BUMP taken as being equal to 1 and the other bits equal to 0.

An advantage of using fuzzy logic is the reduction of the error in the evaluation of distance made by the robot. The sonar usually presents a certain variation. If the variation belongs to a same fuzzy set none misclassification problem occurs. However, a reading error occurs if the variation reaches two neighboring fuzzy sets, i.e., the actual input value belongs to a certain fuzzy set but the sonar reading takes this input signal as belonging to another neighboring fuzzy set.

A set of four actions was defined, being they: **forward, spin right, spin left** and **backward**. The values of rewards are represented in Table 2. To avoid local maximum a rate of 90%-greedy is being used for the choice of the actions. With the environment state defined,

Table 2: Values of rewards

Actions	**Rewards**
Forward	100
Spin Right	50
Spin Left	50
Backward	-50
Bump	-700

a modification of the R-learning algorithm, R'-learning, is proposed in the Fig. 6. This algorithm was tested with Pioneer 1 robot and the results were compared to those obtained by using R-learning algorithm in the next section. Since this modification is performed on the sonar readings and on the immediate reward value, this modification can be used in other reinforcement learning algorithms.

Set ρ and $R(s,a)$, for all s, a arbitrarily
Repeat forever
  $s \leftarrow$ current state
  Choose action $a \in A(s)$ with $\epsilon_r$greedy policy
  Take action a, receive the r value
  Receive the state s' of the fuzzy function
  Receive the certainty degree, φ, of the fuzzy function
  $r' \leftarrow \varphi r$
  $R(s,a) \leftarrow R(s,a) + \alpha[r' - \rho + max_{a'} R(s',a') - R(s,a)]$
  if $R(s,a) = max_a R(s,a)$ then
    $\rho \leftarrow \rho + \beta[r' - \rho + max_{a'} R(s',a') - max_a R(s,a)]$

Figure 6: R'-learning Algorithm

4 Discussion of the Results

Both algorithms, R-Learning and R'-learning, were tested using the Pioneer 1 robot simulator, adopting $\alpha = 0.2$ and $\beta = 0.001$. To establish the first notions of behavior, according to [8], during the first 2000 steps, the robot is controlled by keyboard and the other 8000 with the training of each algorithm.

The performance of reinforcement learning method is sensitive to exploration and reward system. In this work, the strategy of exploration is used to verify what it was already learned, besides avoiding the maximum local. The strategy of exploration uses random movements at rate 10% or 90%-greedy.

Fig. 7 and Fig. 8 show the learning graphs for the R-learning algorithm and R'-learning, respectively. Fig. 9 shows the performance of the learning for both algorithms, where can be noted that the number of collisions obtained through R'-learning algorithm is less than these obtained using R-learning algorithm.

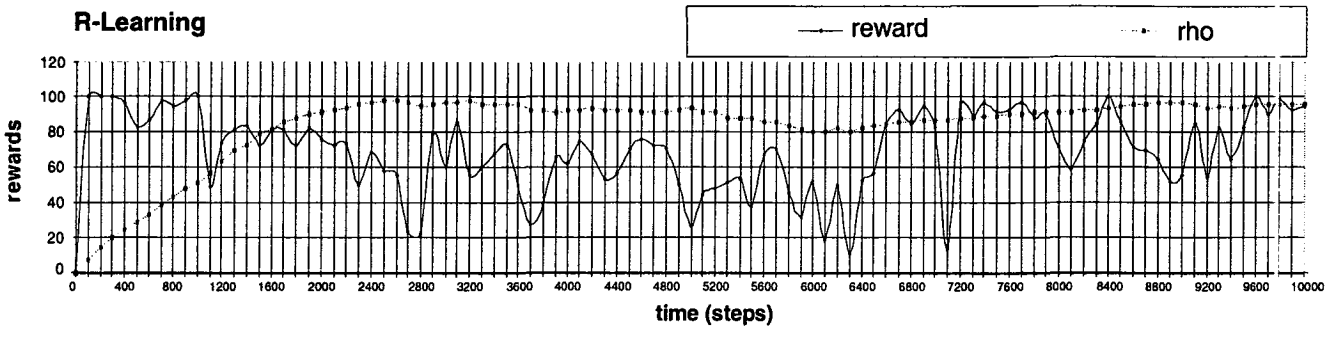

Figure 7: R-Learning without fuzzy logic.

5 Conclusion and Future Work

In this work, a modification of the algorithm R-learning, R'-learning, was proposed for sensor-based navigation of mobile robots. This algorithm incorporates fuzzy logic as for classifying the sonar readings well as for weighting the immediate reward. Since that this modification is

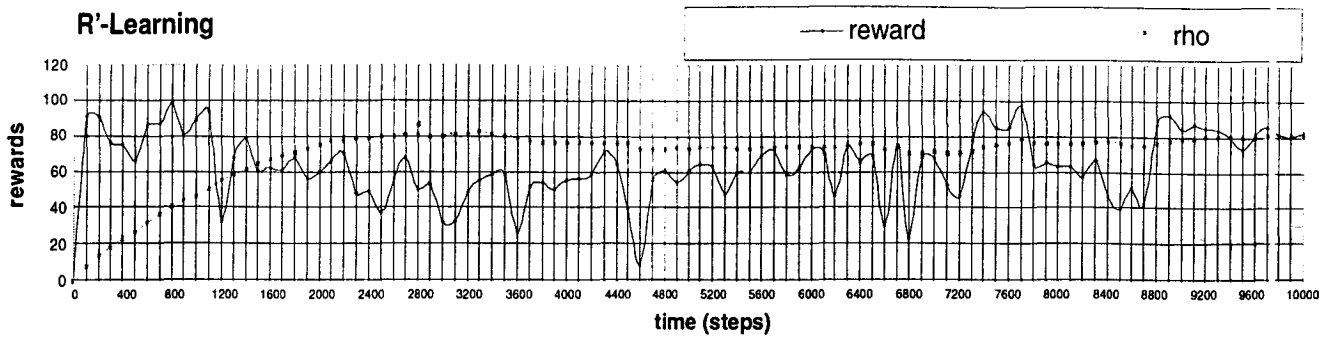

Figure 8: R-Learning using fuzzy logic.

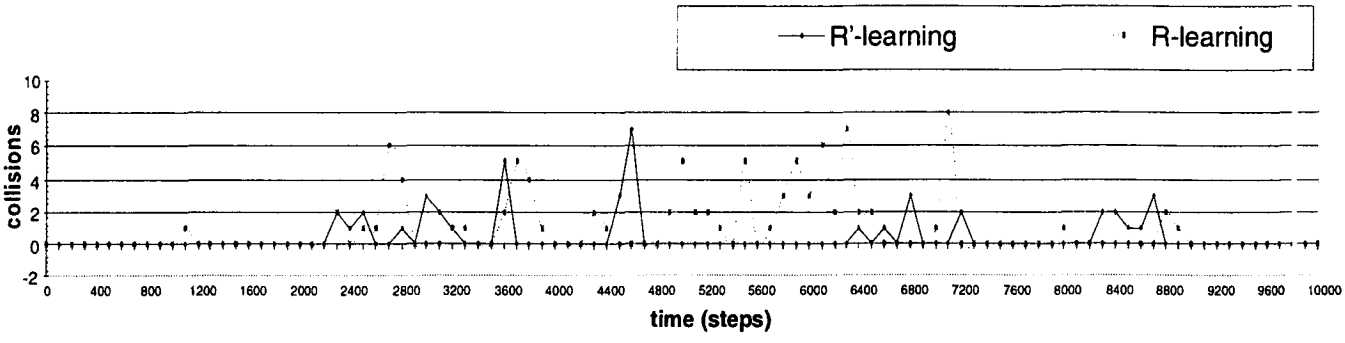

Figure 9: Collisions during the learning.

performed on the sonar inputs and on the immediate reward, this modification can be used in other reinforcement learning algorithms. The results obtained by using R'-learning algorithm were better than those obtained by the R-learning algorithm. However, there are several open questions to be considered. The first one is about the treatment of the error of sonar reading. An initial idea would be to consider a punishment in the immediate reinforcement obtained by each sonar reading that has a larger variation. Another idea is to consider the degree of variation of the sonar reading as a new fuzzy set for which a environment state could belong. A second question is related to the number of signals considered. In this work, the five front sonars were considered as a single signal and so the robot received only 3 input signals. However, we intend to treat the 7 sonar readings independently, because we believe that approach proposed here can be extended for mobile robots having much more sonar sensors.

6 Acknowledgment

The first author would like to acknowledge the financial support from CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil).

References

[1] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[2] H.R. Beon and H.S Cho, "A sensor-based navigation for a mobile robot using fuzzy-logic and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, 1995.

[3] N.H.C Yung and C. Ye, "An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, no. 2, pp. 314–321, 1999.

[4] Richard Bellman, *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

[5] M. L. Puterman, *Markov Decison Process–Discrete Stochastic Dynamic Programming*, Inc. John Wiley & Sons, New York, NY, 1994.

[6] R. A. Howard, *Dynamic Programming and Markov Process*, The MIT Press, Cambridge, MA, 1960.

[7] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Pentice-Hall, Englewood Cliffs, NJ, 1987.

[8] L.P. Kaelbing and M.L. Littman, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[9] Christopher J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. thesis, University of Cambridge, 1989.

[10] Christopher J. C. H. Watkins and Peter Dayan, "Technical note: Q learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[11] Anton Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *Machine Learning: Proceedings of the Tenth International Conference*, San Mateo, CA, 1993, Morgan Kaufmann.

[12] Sridhar Mahadevan, "To discount or not to discount in reinforcement learning: A case study comparing r-learning and q-learning," in *Proceedings of the Eleventh International Conference on Artificial Intelligence*, Morgan Kaufmann, Ed., San Francisco, CA, 1994, pp. 164–172.

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

049/2000 OLIVEIRA, P.R.; ROMERO, R.F.; NONATO, L.G.; MAZUCHELI, J. – Techniques for image compression: a comparative analysis.

048/2000 LINS, L.; LINS, S.; MORABITO, R. – Na n-tet graph approach for non-guillotine packings of n-dimensional boxes into na n-container.

047/2000 OLIVEIRA JR., O. N.; MARCHI, A.R.; MARTINS, M.S.; MARTINS, R.T.; NUNES, M.G.V. – A critical analysis of the performance of english-portuguese-english MT systems.

046/2000 SOUZA, P.S.L.; SANTANA, M.J.; SANTANA, R.H.C. – Escalonamento de processos: uma contribuição para a convergência da área.

045/2000 MARTINS, R.T.; RINO, L.H.M.; NUNES, M.G.V.; MONTILHA, G.; OLIVEIRA JR., O. N. – An interlingua diming at communication on the Web: how language-independent can it be?

044/99 FELTRIM, V.D.; FORTES,R.P.M. – Evaluation of a reverse engineering method through the application in a hypermedia system.

043/99 PANSANATO, L.T.E.; NUNES, M.G.V. – EHDM: Método para projeto de hiperdocumentos para ensino.

042/99 MORABITO, R.; ARENALES, M. – Optimizing the cutting of stock plates in a furniture company.

041/98 SANTOS-MEZA, E.; SANTOS, M.O.; ARENALES, M.N. – Lot sizing and scheduling in na automated foundry.

040/98 FELTRIM, V.D.; FORTES, R.P.M. – Uma modelagem do domínio de engenharia reversa de software utilizando o método OOHDM.