

A DISTRIBUTED REINFORCEMENT LEARNING CONTROL ARCHITECTURE FOR MULTI-LINK ROBOTS

Experimental Validation

Jose Antonio Martin H.

Dep. Sistemas Informáticos y Computación, Universidad Complutense de Madrid
C. Prof. José García Santesmases, 28040, Ciudad Univesitaria, Madrid, Spain

Javier de Lope

Dept. Applied Intelligent Systems, Universidad Politécnica de Madrid
Campus Sur, Ctra. Valencia, km. 7, 28031 Madrid, Spain

Keywords: Reinforcement Learning, Multi-link Robots, Multi-Agent systems.

Abstract: A distributed approach to Reinforcement Learning (RL) in multi-link robot control tasks is presented. One of the main drawbacks of classical RL is the combinatorial explosion when multiple states variables and multiple actuators are needed to optimally control a complex agent in a dynamical environment. In this paper we present an approach to avoid this drawback based on a distributed RL architecture. The experimental results in learning a control policy for diverse kind of multi-link robotic models clearly shows that it is not necessary that each individual RL-agent perceives the complete state space in order to learn a good global policy but only a reduced state space directly related to its own environmental experience. The proposed architecture combined with the use of continuous reward functions results of an impressive improvement of the learning speed making tractable some learning problems in which a classical RL with discrete rewards (-1,0,1) does not work.

1 INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto, 1998) is a paradigm of Machine Learning (ML) that consists in guiding the learning process by rewards and punishments.

One of the most important breakthroughs in RL was the development of Q-Learning (Watkins, 1989). In a simple form its learning rule is:

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_a Q(s_{t+1}, *) - Q(s, a)] \quad (1)$$

Note that this is the classical *moyenne adaptive modifiée* technique (Venturini, 1994) which adaptively approximates the average μ of a set $x = (x_1 \dots x_\infty)$ of observations:

$$\mu = \mu + \alpha[x_t - \mu], \quad (2)$$

thus replacing the factor x_t in (2) with the current reward r plus a fraction $\gamma \in (0, 1]$ of the best possible approximation to the maximum future expected reward $\max_a Q(s_{t+1}, *)$:

$$\mu = \mu + \alpha[r + \gamma \max_a Q(s_{t+1}, *) - \mu], \quad (3)$$

we are indeed approximating the average of the maximum future reward when taking the action a at the state s .

In (Watkins and Dayan, 1992) the authors proved convergence of Q-Learning to an optimal control policy with probability 1 under certain single assumptions see (Sutton and Barto, 1998, p.148).

In this paper we present a distributed approach to RL in robot control tasks. One of the main drawbacks of classical RL is the combinatorial explosion when multiple states variables and multiple actuators are needed to optimally control a complex agent in a complex dynamical environment. As an example, a traditional RL system for the control of a robot with multiple actuators (i.e. multi-link robots, snake-like robots, industrial manipulators, etc.) when the control task relies on feedback from the actuators. This situation produces a huge table of *states* \times *actions* pairs which, in the best case, assuming enough memory to store the table, the learning algorithm needs a huge amount of time to converge, indeed the problem could become intractable.

Also, for some nonlinear parameterized function approximators, any temporal-difference (TD) learning method (including Q-learning and SARSA) can become unstable (parameters and estimates going to infinity) (Tsitsiklis and Roy, 1996). For the most part, the theoretical foundation that RL adopts from dynamic programming is no longer valid in the case of function approximation (Sutton and Barto, 1998). For these reasons the development of distributed algorithms relying on partial observations of states and individual action selections mechanisms becomes crucial to solve such intractable problems.

Our approach avoids this drawback based on a distributed architecture. The work is based on the hypothesis that such complex agents can be modeled as a colony of single interacting agents with single state spaces as well as single action spaces. This hypothesis is verified empirically by the means of clear experimental results showing the viability of the presented approach.

The rest of the paper is organized as follows. First we introduce some previous works on RL for robot controlling. In the next section we present the basis of our distributed RL control approach, including a pseudo-code of the proposed algorithm. Next we define an application framework in which we have carried out the experiments. For the experiments we use two multi-link robot models, a three-link planar robot and an industrial SCARA.

2 RL IN ROBOT CONTROL

RL has been applied successfully to some robot control task problems (Franklin, 1988; Lin, 1993; Mataric, 1997; Rubo et al., 2000; Yamada et al., 1997; Kretchmar, 2000; Kalmar et al., 2000; El-Fakdi et al., 2005) and is one of the most promising paradigms in robot learning. This is mainly due to its simplicity. The main advantages of RL over other methods in robot learning and control are:

1. There is no need to know a prior model of the environment. This is a crucial advantage because in most complex tasks a model of the environment is unknown or is too complex to describe.
2. There is no need to know previously what actions for each situation must be presented to the learner.
3. The learning process is online by directly interacting with the environment.
4. It is capable of learning from scratch.

RL can be applied to robot control tasks in a very easy way. The basic elements of a RL learning system are:

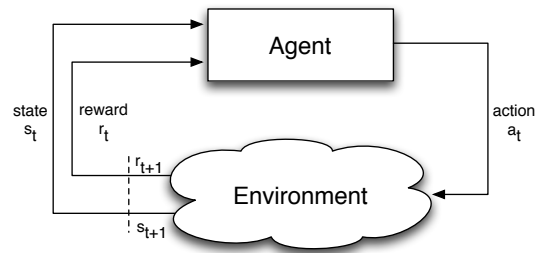


Figure 1: The basic model of an RL system.

1. An agent or a group of agents (i.e. mobile robots) that perceives the environment (typically called state S_t) and behaves on the environment producing actions (a_t).
2. The environment in which the agents live, that could be a simulated world or the real world.
3. A reward signal r_t that represents the evaluation of the state and it is used by the agents to evaluate its behavioral policies.

Fig. 1 shows the agent-environment interaction in RL. The control cycle can be used directly to control any robotic task where the agent represents a robot.

The majority of RL algorithms rely on the condition that the state signal S_t fulfills the Markov property that is, the signal S_t must contain for the time step (t) all the information needed to take an optimal action at time step (t). In other words we can define that a signal state S_t fulfills the Markov property if and only if there is no information that could be added into S_t that could produce a more optimal action than the original signal control.

We use SARSA as the basis for our RL algorithm. SARSA has been proved to converge under the same set of single assumptions that Q-Learning (Sutton and Barto, 1998) but empirically it outperforms the Q-Learning's performance. The pseudo-code of the SARSA algorithm is described in Fig. 2 which uses a slight variation of the learning rule shown in (1).

3 PROPOSED ARCHITECTURE

In this work we propose a distributed architecture for RL in order to prevent the combinatorial explosion when multiple states and actuators are needed. The main feature of this architecture is the division of the learning problem into different agents each one for every different actuator (i.e. joint or motor). Fig. 3 shows the basic control cycle of this architecture.

```

Initialize  $Q$  arbitrarily
Repeat( for each episode )
  Observe  $s$ 
  Select  $a$  for  $s$  by  $\epsilon$ -greedy policy
  Repeat( for  $n$  steps )
    Take action  $a$ , observe  $r, s'$ 
    Select  $a'$  for  $s'$  by  $\epsilon$ -greedy policy
     $Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
     $s = s'; a = a'$ 
  until  $s$  is terminal
    
```

Figure 2: SARSA: RL control algorithm.

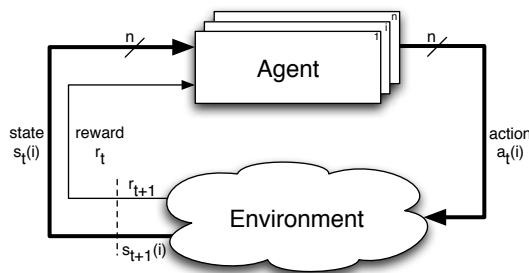


Figure 3: The basic model of the Distributed Approach to RL.

In Fig.3 we can see that the RL signal r_t is the same for all agents and the state signal s_{t+1} is propagated against all agents despite each individual agent can filter it and use only the information needed to behave properly. Also, we must note that each agent emits a different action to the environment. Thus, each agent will have its own independent memory in which to store its respective knowledge and hence the quantity of state-action pairs in the global system is considerably less than within a full state-action space due to the additive scheme instead of the multiplicative scheme which is the main cause of the curse of dimensionality. Fig. 4 shows the proposed algorithm based on the SARSA algorithm which reflects the proposed multi-agent control architecture.

Within this distributed approach every agent will have its own Q -table Q_i and its own perceptions $f_i(s_i)$. In the case of multi-link robots, each agent will perceive as its own state signal only the current state of its respective actuator plus the information of the goal reaching index.

3.1 Continuous Rewards

Traditionally RL systems work with discrete rewards (i.e. [-1, 0, 1]) but this is not mandatory due to the algorithms actually can handle scalar reward signals.

```

Initialize  $Q_i$  arbitrarily ;  $\forall i = 1, \dots, n$ 
Repeat( for each episode )
  Observe  $(s_1, \dots, s_n)$ 
  Select  $a_i$  for  $s_i$  by  $\epsilon$ -greedy policy ;  $\forall i = 1, \dots, n$ 
  Repeat( for  $J$  steps )
    Take actions  $a_1, \dots, a_n$ , observe  $r, s'_1, \dots, s'_n$ 
    Select  $a'_i$  for  $s'_i$  by  $\epsilon$ -greedy policy ;  $\forall i = 1, \dots, n$ 
     $Q_i(s_i, a_i) = Q_i(s_i, a_i) + \alpha[r + \gamma Q_i(s'_i, a'_i) - Q_i(s_i, a_i)]$ 
     $s_i = s'_i; a_i = a'_i$ 
  until  $s$  is terminal
    
```

Figure 4: MA-SARSA: A Multi-agent RL control algorithm.

Indeed the conditions for proper convergence assume scalar rewards signals. In our model we use continuous rewards functions as a way to improve convergence and learning speed. The use of continuous rewards signals may lead to some local optima traps but the exploratory nature of RL algorithms tends to escape from these traps.

One elegant way to formulate a continuous reward signal in the task of reaching a goal is simply to use the inverse Euclidean distance plus 1 as shown in (4):

$$reward = \frac{\beta}{1 + d^n}, \quad (4)$$

where β is a scalar that controls the absolute magnitude of the reward, d is the Euclidean distance from the goal position and n is an exponent which is used to change the form in which the function responds to changes in the Euclidean distance. In our model we use generally (4) as the reward when the state is terminal and the goal is reached otherwise we use another continuous function as a penalty function for every action that does not reach the goal as shown in (5).

$$punishment = -\beta d^n - \epsilon, \quad (5)$$

where the term ϵ is used to include some constraints that the system engineer considers appropriate.

We have used this same kind of continuous rewards functions in past works with successful results for multi-objective RL systems (Martin-H. and De-Lope, 2006).

4 EXPERIMENTAL APPLICATION FRAMEWORK

Experiments have been realized on two multi-link robot models: a three-link planar robot and a SCARA robot. The experiments are made with the intention that the robots learn their inverse kinematics in order

to reach a desired configuration in the joint and Cartesian spaces. For this reason we have selected both types of manipulators.

4.1 Three-Link Planar Robot

The three link planar robot is a configuration which is generally studied by the Robotics community and its kinematics is well-known. It includes some redundancy in the calculus of the inverse kinematics by means of analytic procedures. For this reason, it is very interesting for the verification of our methods in this kind of mechanical structures. As we have commented previously the same approach could be applied for controlling the movement and body coordination of snake-like robots.

Fig. 5 shows a diagram of the robot model¹. Its Denavit-Hartenberg parameters are described in the Table 1 and its direct kinematic equation is defined in (6).

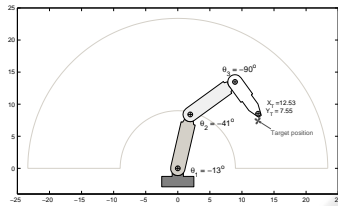


Figure 5: Three link-planar robot model.

Table 1: Three link planar robot D-H parameters.

i	θ_i	d_i	a_i	α_i
1	θ_1	0	$a_1 = 8.625$	0
2	θ_2	0	$a_2 = 8.625$	0
3	θ_3	0	$a_3 = 6.125$	0

$$T^3 = \begin{pmatrix} C_{123} & -S_{123} & 0 & a_1 C_1 + a_2 C_{12} + a_3 C_{123} \\ S_{123} & C_{123} & 0 & a_1 S_1 + a_2 S_{12} + a_3 S_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (6)$$

where S_{123} and C_{123} correspond to $\sin(\theta_1 + \theta_2 + \theta_3)$ and $\cos(\theta_1 + \theta_2 + \theta_3)$, respectively (and equally for S_1, S_{12}, C_1 and C_{12}), θ_1, θ_2 and θ_3 define the robot joint coordinates — θ_1 for the first joint, located near

¹The original model is by Matt Kontz from Walla Walla College, Edward F. Cross School of Engineering, February 2001.

to the robot base, θ_2 for the middle joint and θ_3 for the joint situated in the final extreme— and a_1, a_2 and a_3 correspond to the physical length of every link, first, second and third, respectively, numbering from the robot base.

4.2 Scara Robot

The SCARA robot was selected due to it is a widely used robot in the industry. It has well-known properties for the use as manufacturing cells assisting conveyor belts, electronic equipment composition, welding tasks, and so on. In this case we are using a real three dimension robot manipulator so the agents have to try to reach an objective configuration in a 3D space being able to generate an approaching real-time trajectory when the agent system is completely trained.

A physical model of the SCARA robot is shown in Fig. 6. The Denavit-Hartenberg parameters are defined in Table 2 and the direct kinematic matrix is shown in (7).

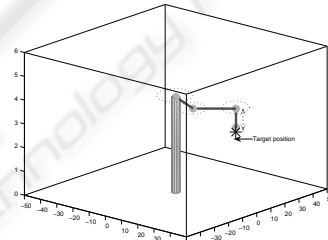


Figure 6: SCARA robot model.

Table 2: SCARA robot D-H parameters.

i	θ_i	d_i	a_i	α_i
1	θ_1	0	$a_1 = 20$	0
2	θ_2	0	$a_2 = 20$	π
3	θ_3	0	0	0
4	0	d_4	0	0

$$T^4 = \begin{pmatrix} C_{12-3} & -S_{12-3} & 0 & a_1 C_1 + a_2 C_{12} \\ S_{12-3} & C_{12-3} & 0 & a_1 S_1 + a_2 S_{12} \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (7)$$

where S_{12-3} and C_{12-3} correspond to $\sin(\theta_1 + \theta_2 - \theta_3)$ and $\cos(\theta_1 + \theta_2 - \theta_3)$, respectively (and equally for S_1, S_{12}, C_1 and C_{12}), $\theta_1, \theta_2, \theta_3$ and d_4 are the joint parameters for the shoulder, elbow, wrist and prismatic joints, respectively, and a_1 and a_2 the lengths of the arm links.

4.3 Experiments Design

The control task consists of to reach a continuously changing goal position of the robot end-effector by means of a random procedure. The goal positions are defined in such a way that they are always reachable for the robot. Thus, the learning process needs to develop an internal model of the inverse kinematics of the robot which is not directly injected by the designer in a *a priori* way. Through the different trials, a model of the robot inverse kinematics is *learned* by the system.

When a goal position is generated, the robot tries to reach it. Each trial can finish as a success episode, i.e. the robot reaches the target at a previously determined time or as a fail episode, i.e. the robot is not able to adopt a configuration to reach the goal position. In both cases the system parameters are updated using the previously defined method and a new goal position is randomly generated.

Once the systems parameters have been tuned by the RL process, the system has learned an inverse kinematics model of the robot. Such a model can be used for real time control of the robot. The learning process duration depends on the initial conditions and the randomly generated positions since RL relies on a kind of stochastic search, but as it is shown in the experimental results this time is very short. We will comment in detail on the results in the next section.

5 EXPERIMENTAL RESULTS

The test procedure defined above has been applied to both robots the three-link planar robot and the SCARA robot. Although the results are very similar we offer separate results for each robot.

5.1 The Three-link Planar Robot

The three-link planar robot has a relatively simple inverse kinematics, although its three rotation axes are in parallel, generating multiple candidate configurations for the same position and orientation. The learning system produces extremely good results, finding a model of the inverse kinematics in 50 episodes in average, being the best case of 4 episodes. The average number of episodes is obtained by a sequence of 30 multiple restarting experiments and calculating the point at which the convergence is achieved for each run.

Fig. 7 shows the learning curve for this best case. As we can observe, before the system runs 4 episodes, the controller converges to the final solution. The

peaks displayed are due to the randomly nature of the experiment: the vertical axis represents the steps that were needed to reach the goal, if the new randomly generated goal is far from the current position, then more steps will be needed in order to reach the goal position.

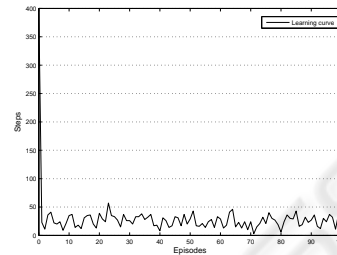


Figure 7: Learning curve for the three-link planar robot model for 100 episodes.

Fig. 8 shows a trace of the behavior of the three-link planar robot for four consecutive goals. It can be seen that there is some little noise in the trajectory in some segments of the paths between consecutive goals. This noise is the product of the nature of the goal reaching information due to some zero-crossings at the orthogonal axes (x,y), this must be interpreted as search for accuracy in the goal reaching attempt. We must recall that we are using a mixture of memory based approach and a reactive mechanism thus these oscillations are a natural consequence of a reactive behavior. This noise can be reduced by some methods to prevent oscillations in which we are working as a future improvement.

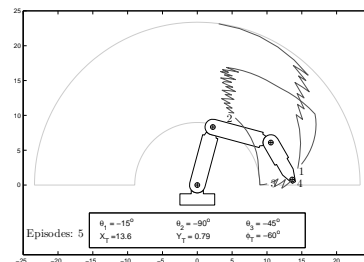


Figure 8: Trace for the three-link planar robot for four consecutive goals.

5.2 The Scara Robot

The SCARA robot presents very similar results. Fig. 9 shows a learning curve for 100 episodes. In these cases the learning process converges slower than the one of previous robot. The abrupt peaks that appears in the first episodes show the adaptation pro-

cess (learning). For the SCARA robot 67 episodes are needed as average to find a solution, obtaining a best case after 23 episodes.

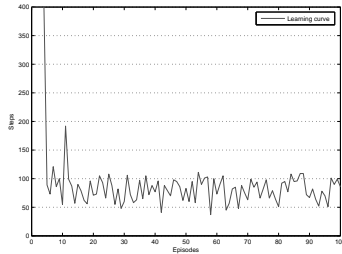


Figure 9: Learning curve for the SCARA robot model for 100 episodes.

Fig. 10 shows a trace of the behavior of the SCARA robot for four consecutive goals, in this picture the same oscillations can be seen as in the trace of the three-link planar robot due to the zero-crossing effect explained above.

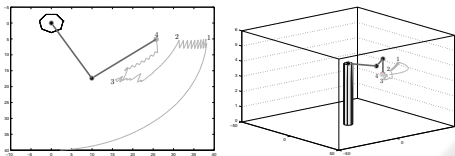


Figure 10: Trace for the SCARA robot for four consecutive goals.

6 CONCLUSIONS

A distributed approach to RL in robot control tasks has been presented. To verify the method we have defined an experimental framework and we have tested it on two well-known robotic manipulators: a three-link planar robot and an industrial SCARA.

The experimental results in learning a control policy for diverse kind of multi-link robotic models clearly shows that it is not necessary that the individual agents perceive the complete state space in order to learn a good global policy but only a reduced state space directly related to its own environmental experience. Also we have shown that the proposed architecture combined with the use of continuous reward functions results in an impressive improvement of the learning speed making tractable some learning problems in which a classical RL with discrete rewards $(-1, 0, 1)$ does not work. Also we want to adapt the proposed method to snake-like robots. The main drawback in this case could be the absence of a base which fixes the robot to the environment and its implication in the definition of goal positions.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Ministry of Science and Technology, project DPI2006-15346-C03-02.

REFERENCES

- El-Fakdi, A., Carreras, M., and Ridao, P. (2005). Direct gradient-based reinforcement learning for robot behavior learning. In *ICINCO 2005*, pages 225–231. INSTICC Press.
- Franklin, J. A. (1988). Refinement of robot motor skills through reinforcement learning. In *Proc. of 27th Conf. on Decision and Control*, pages 1096–1101, Austin, Texas.
- Kalmar, Szepesvari, and Lorincz (2000). Modular reinforcement learning: A case study in a robot domain. *ACTACYB: Acta Cybernetica*, 14.
- Kretchmar, R. M. (2000). *A synthesis of reinforcement learning and robust control theory*. PhD thesis, Colorado State University.
- Lin, L.-J. (1993). Scaling up reinforcement learning for robot control. *Machine Learning*.
- Martin-H., J. A. and De-Lope, J. (2006). Dynamic goal coordination in physical agents. In *ICINCO 2006*, pages 154–159. INSTICC Press.
- Mataric, M. J. (1997). Reinforcement learning in the multi-robot domain. *Auton. Robots*, 4(1):73–83.
- Rubo, Z., Yu, S., Xingoe, W., Guangmin, Y., and Guochang, G. (2000). Research on reinforcement learning of the intelligent robot based on self-adaptive quantization. In *Proc. of the 3rd World Congr. on Intelligent Control and Automation. IEEE, Piscataway, NJ, USA*, volume 2, pages 1226–9.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning, An Introduction*. The MIT press.
- Tsitsiklis, J. N. and Roy, B. V. (1996). Analysis of temporal-difference learning with function approximation. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *NIPS*, pages 1075–1081. MIT Press.
- Venturini, G. (1994). Adaptation in dynamic environments through a minimal probability of exploration. In *SAB94*, pages 371–379, Cambridge, MA, USA. MIT Press.
- Watkins, C. J. (1989). *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, Cambridge, United Kingdom.
- Watkins, C. J. and Dayan, P. (1992). Technical note Q-learning. *Machine Learning*, 8:279.
- Yamada, S., Watanabe, A., and Nakashima, M. (1997). Hybrid reinforcement learning and its application to biped robot control. In *NIPS*. The MIT Press.