Decentralized Reinforcement Learning Applied to Mobile Robots

David L. Leottau[1], Aashish Vatsyayan[2],
Javier Ruiz-del-Solar[1], and Robert Babuška[2]

[1] Advanced Mining Technology Center, Department of Electrical Engineering, Universidad de Chile, Av. Tupper 2007, Santiago, Chile
[2] Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands

Abstract. In this paper, decentralized reinforcement learning is applied to a control problem with a multidimensional action space. We propose a decentralized reinforcement learning architecture for a mobile robot, where the individual components of the commanded velocity vector are learned in parallel by separate agents. We empirically demonstrate that the decentralized architecture outperforms its centralized counterpart in terms of the learning time, while using less computational resources. The method is validated on two problems: an extended version of the 3-dimensional mountain car, and a ball-pushing behavior performed with a differential-drive robot, which is also tested on a physical setup.

Keywords: multiagent learning, decentralized control, reinforcement learning, robot soccer.

1 Introduction

Reinforcement learning (RL) has been increasingly used to learn complex behaviors for robots in the real world [1, 2]. One of the main challenges is the large number of training trials required, especially in systems with many state and action variables [3]. For such problems, distributed reinforcement learning can be used to address this issue [4]. For instance, in mobile robotics, a common high-level motion command is the desired velocity vector (e.g.: $[v_{right}, v_{left}]$ for a differential robot, or $[v_x, v_y, v_\theta]$ for an omnidirectional robot). If each component of this vector is handled individually, a distributed control scheme can be applied. By taking care of the coordination among the agents, it is possible to use decentralized methods [5] to learn behaviors which require these motion commands, taking an advantage of parallel computation and other benefits of multiagent systems (MAS) [6, 4].

In Decentralized Reinforcement Learning (DRL), a problem is decomposed in several learning tasks, or sub-problems, whose information and resources are managed separately and these tasks work together toward a common goal. In systems with multidimensional action spaces, each individual action variable is handled by a separate agent.

In this paper we propose to use DRL for mobile robots, where each component of the desired velocity vector (e.g.: $[v_l, v_w]$, linear and angular speed for the particular case of a differential-drive robot) is learned by a separate agent. Since most of the MAS works reported in the literature do not address or validate MultiAgent Learning (MAL) algorithms with multiple-state, stochastic, and real world problems [4], our goal is to show that MAS are also applicable to real-world problems like robotic platforms, by using a DRL architecture. Thereby, two separate problems are considered. The first is an extended version of the three dimensional mountain car (3DMC) [7], which is a common RL test bed. The second is a ball-pushing behavior, a soccer task performed with a differential-drive robot in a noisy and stochastic setting, which is also tested with a physical setup. Both validation problems are modeled and implemented by using a Centralized RL (CRL) and a DRL architecture, in order to compare and analyse both approaches.

The main contribution of this paper is twofold: first, we propose a DRL scheme for learning individual behaviors in the context of mobile robots; second, we compare CRL with DRL on two different validation problems. To the best of our knowledge, this is the first decentralized architecture for learning on mobile robot platforms, along with a comparison with the centralized RL counterpart.

The remainder of this paper is organized as follows: Section 2 gives a brief introduction to DRL, Section 3 introduces the control problems, Section 4 presents and analyses the experimental results, Section 5 presents the related work, and Section 6 concludes the paper and outlines future work.

2 Decentralized Reinforcement Learning

In DRL, the learning problem is decomposed into several sub-problems which as learned in parallel by separate agents. The MAS perspective yields several potential advantages if the problem is approached with decentralized learners and the coordination is taken care of [4]:

- The learning speed might be higher compared to a centralized agent which has to search an exponentially larger action space.
- The state space can be reduced if not all the state information is relevant to all the learning agents.
- Different algorithms, models or configurations can be used independently by the different agents.
- Memory and processing time requirements are smaller.
- Parallel or distributed computing implementations are suitable.

A DRL scheme also has several challenges which must be efficiently solved in order to take advantage of aforementioned MAS benefits. Agents have to coordinate their individual behaviors towards a coherent and desired joint behavior. The formulation of a good DRL modeling and learning goal is a difficult problem [4].

3 Validation Problems

In order to validate the DRL approach, two different problems have been selected: the 3-Dimensional mountain car (3DMC), a canonical and already reported RL test-bed [7]; and the ball-pushing behavior, a noisy and stochastic real world application, which is performed with the MiaBotPro [8] differential-drive robot, and tested on a physical setup. These problems allow us to carry out a comparative analysis between a DRL scheme and its CRL counterpart. For the case of DRL implementations, both problems are modeled with two RL agents, no explicit coordination mechanism, or MAL algorithm; so, indirect coordination will emerge between the two independently learning agents.

3.1 Three-Dimensional Mountain Car

Centralized modeling: An under-powered car has to move to its goal state [7]. The slope of the mountain is shown in Figure 1. The state has four continuous-valued features: $x, \dot{x}, y, \dot{y}$. The positions (x, y) have the range of $[-1.2, 0.6]$ and the speeds $(\dot{x}, \dot{y})$ are constrained to $[-0.07, 0.07]$. The agent selects from five actions: {Neutral, West, East, South, North}. West and East modify $\dot{x}$ by -0.001 and +0.001 respectively, while South and North modify $\dot{y}$ by -0.001 and $+0.001$ respectively. On each time step $\dot{x}$ is updated by $0.025(\cos(3x))$ and $\dot{y}$ is updated by $-0.025(\cos(3y))$ due to gravity. The goal state is $x \geq 0.5 and\ y \geq 0.5$. The agent begins at rest at the bottom of the hill. The reward is -1 for each time step until the goal is reached, at which point the episode ends and the reward is 0. The episode also ends, and the agent is reset to the start state, if the agent fails to find the goal within 5000 time steps.

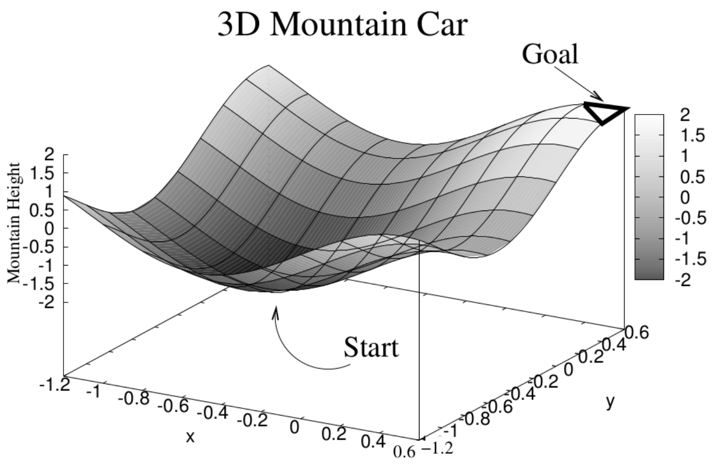


Fig. 1: 3D mountain car surface. Figure adopted from [7].

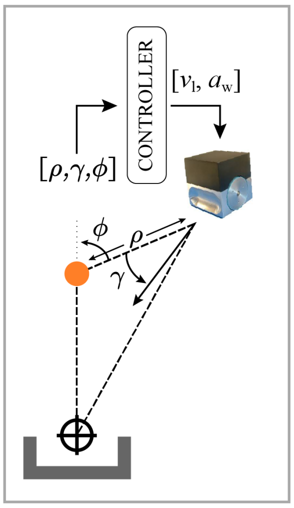
Proposed decentralized modeling: In the original 3D mountain car problem, a centralized approach is followed. The original 5 actions modeling (CRL-5a) make it impossible for the car to turn or perform a diagonal move at each time step. In order to make this problem fully decentralized, more realistic, and challenging, we have extended the action space by incorporating four more actions: {NorthWest, NorthEast, SouthWest, SouthEast}. Since the car is now able to move on x and y axes at the same time, $\dot{x}$, and $\dot{y}$ updates must be multiplied by $1/\sqrt{2}$ for these new four actions because of the diagonal moves. The decentralized approach employs two independent agents: $agent^x$ whose action space is $\{Neutral, West, East\}$, and $agent^y$ whose action space is $\{Neutral, South, North\}$. The learning task then be seen as two independent, parallel sub-tasks, $agent^x$ trying to reach the east top, and $agent^y$ trying to reach the north top.

Performance index: The evolution of the learning process is evaluated by measuring and averaging 25 runs. The performance index is the cumulative reward per episode, where $-5,000$ is the worst case, and zero the best, though unreachable case.

RL algorithm and optimized parameters: SARSA(λ) with Radial Basis Function (RBF) approximation with ϵ-greedy exploration is implemented for these experiments [1]. The exploration rate ϵ is decayed by 0.99 at the end of each learning episode. The following parameters are obtained after the hill climbing optimization procedure: learning rate (α), eligibility traces decay factor (λ), and exploration probability (ϵ). These parameters are detailed in Table 2 for each implemented scheme. Additionally, the number of Gaussian RBF cores per feature were also optimized: 9 cores to x and y, 6 cores to $\dot{x}$ and $\dot{y}$, and a standard deviation per core $1/2 \cdot |feature_{max} - feature_{min}|/nCores$.

A summary of the three implemented cases is shown below:

- *CRL Original model (CRL-5a):*
  Actions: {Neutral, West, East, South, North}
  Global reward function: $r = 0$ if goal, $r = -1$ otherwise
  Joint state vector: $[x, \dot{x}, y, \dot{y}]$
- *CRL Extended model (CRL-9a):*
  Actions: {Neutral, West, NorthWest, North,
  NorthEast, East, SouthEast, South, SouthWest}
  Global reward function: $r = 0$ if goal, $r = -1$ otherwise
  Joint state vector: $[x, \dot{x}, y, \dot{y}]$
- *Decentralized RL model (DRL):*
  Actions $agent^x$: {Neutral, West, East},
  Actions $agent^y$: {Neutral, South, North}
  Individual reward functions:
  $r_x = 0$ if $x \geq 0.5$, $r_x = -1$ otherwise; $r_y = 0$ if $y \geq 0.5$, $r_y = -1$ otherwise
  Joint state vector: $[x, \dot{x}, y, \dot{y}]$



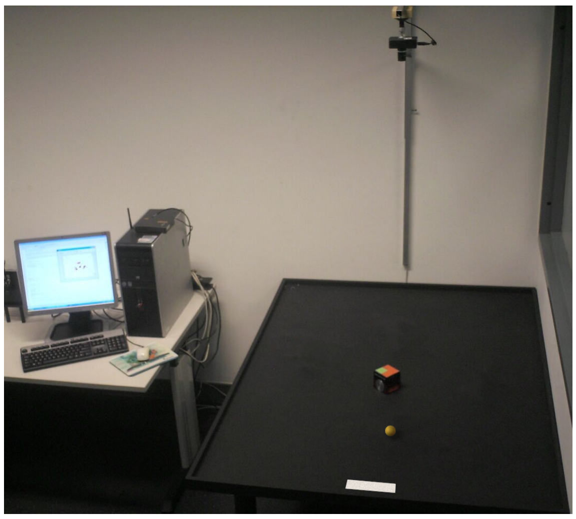


Fig. 2: Definition of variables for the ball-pushing problem (left), and, a picture of the implemented experimental setup (right).

3.2 Ball-Pushing

We are considering the ball-pushing behavior [9], a basic robot soccer skill [2] similar to [10, 11], where a differential robot player attempts to push the ball and score a goal. The MiaRobot Pro is considered for this implementation (See Figure 2. In the case of a differential robot, the complexity of this task comes from its non-holonomic nature, limited motion and accuracy, and especially from the highly dynamic and non-linear physical interaction between the ball and the robot's irregular front shape. The description of the desired behavior will use the following variables: $[v_l, v_w]$, the velocity vector composite by linear and angular speeds; a_w, the angular acceleration; γ, the robot-ball angle; ρ, the robot-ball distance; and, ϕ, the robot-ball-target complementary angle. These variables are shown in Figure 2 at left, where the center of the goal is located in $\oplus$, and a robot's egocentric reference system is considered with the x axis pointing forwards.

RL procedure is carried out episodically. After a reset, the ball is placed in a fixed position $20cm$ in front of the goal, the robot is set on a random position behind the ball and the goal. The successful terminal state is reached if the ball crosses the goal line. If robot leaves the field is also considered a terminal state. The RL procedure is carried out in a simulator and the bests learned policy obtained between the 25 runs for the CRL and DRL implementations are directly transferred and tested on the MiaBot Pro robot on the experimental setup.

Centralized modelling: For this implementation, proposed control actions are twofold $[v_l, a_w]$, the requested linear speed and the angular acceleration, where $A^{a_w} = [positive, neutral, negative]$. Our expected policy is to move fast and push the ball towards the goal. That means: to minimize ρ, γ, ϕ; and to maximize v_l. Thus, this centralized approach considers all possible actions combinations $\mathcal{A} = A^{v_l} \cdot A^{a_w}$ and learns $[v_l, a_w]$ actions, from the observed joint state $[\rho, \gamma, \phi, v_w]$, where $[v_w = v_{w(k-1)} + a_w]$. States and actions are detailed in Table 1.

Table 1: Description of state and action spaces for the DRL modeling of the ball-pushing problem

Joint state space: $S = [\rho, \gamma, \phi, v_w]^T$			
Feature	**Min.**	**Max.**	**N.Cores**
ρ	0 mm	1000 mm	5
γ	-45deg	45deg	5
ϕ	-45deg	45deg	5
v_w	-10deg/s	10deg/s	5
Decentralized action space: $A = [v_l, a_w]$			
Agent	**Min.**	**Max.**	**N.Actions**
v_l	0 mm/s	100 mm/s	7
a_w	-2 deg $/s^2$	2 deg $/s^2$	3
Centralized action space: $A = [v_l \cdot a_w]$			
$A_T = A^{v_l} \cdot A^{a_w} = 5 \cdot 3 = 15$ actions			

Decentralized modelling: Differential robot velocity vector can be split in two independent actuators, right and left wheel speeds $[v_r, v_l]$ or linear and angular speeds $[v_l, v_w]$. To keep parity with the centralized model, our decentralized modelling considers two single agents for learning v_l and a_w in parallel as it is depicted in Table 1. In this way, the *ball-pushing* behavior can be decomposed in two sub-task, *ball-shooting* and *ball-goal-aligning*, which are performed respectively by $agent^{vl}$ and $agent^{a_w}$. The joint state vector $[\rho, \gamma, \phi, v_w]$ is identical to the one proposed for the centralized case.

A common reward function is considered for both CRL and DRL implementations, it is shown in expression (1), where P_{bg} is the distance where the ball crossed goal line with respect to the center of the goal, $gpSize = 75\ mm$ is the distance from the center of the goal to the post, $K = 10$ is a constant gain, and max features are normalization values taken from Table 1.

$$R(s) = \begin{cases} K \cdot (1.1 - |P_{bg}|/(gpSize)) & \text{if goal} \\ -(\rho/\rho_{max} + \gamma/\gamma_{max} + \phi/\phi_{max}) & \text{otherwise} \end{cases} \quad (1)$$

Performance index: The evolution of the learning process is also evaluated by measuring and averaging 25 runs. Percentage of scored goals across trained episodes is considered as performance index:
$\%ofScoredGoals = scoredGoals/Episode$, where $scoredGoals$ are the amount of scored goals until the current training $Episode$. Final performance is also measured running again a thousand episodes with the best policy (between 25) obtained per each tested scheme.

RL algorithm and optimized parameters: A RBF SARSA(λ) algorithm with softmax action selection is implemented for these experiments [1]. Boltzman exploration temperature is decayed as:
$\tau = \tau_0 \cdot exp(-dec \cdot episode/maxEpisodes)$, where $episode$ is the current episode index and $maxEpisodes = 1000$ trained episodes per run. Thereby, the following parameters are optimized: learning rate (α), eligibility traces decay factor (λ), Boltzman exploration initial temperature (τ_0), and exploration decay factor (dec). Obtained values after optimization are listed in Table 1. Furthermore, number of discretized actions for the linear velocity are optimized obtaining $A^{v_l} = 5$ for the CRL scheme and $A^{v_l} = 7$ for the DRL.

4 Experimental Results and Analysis

4.1 Three-Dimensional Mountain Car

Figure 3 shows a performance comparison between: the original implementation of 3DMC proposed at [7], CRL-5a; the extension of that original problem where 9 actions are considered, CRL-9a; and a decentralized scheme, DRL; It is important to remember that a better performance tends from negative to zero. Table 2 shows the averaged final performance of the last 100 episodes. Our results for CRL-5a (red dotted line in Figure 3) converges considerably faster than results presented in [7], it can be due to parameter optimization, and because we have implemented a RBF approach instead CMAC for continuous states generalization. CRL-9a converges slower than the original one as it is expected because of the augmented action space. Notice that DRL speeds-up convergence and outperforms both centralized schemes. From error bars in Figure 3, it can be noticed that asymptotic performance is similar between the three implementations. Thereby, the most noticeable result is the fact that the DRL scheme is able to learn faster than CRL ones without loosing performance. Expressing learning speed as a *time to threshold* as it is presented in Table 2, DRL is two times faster than CRL-5a and it has a better performance in around 15 units. Further, DRL is almost three times faster than CRL-9a, its direct centralized counterpart, showing an outperform of around 35 units.

Regarding computational resources, from optimized parameters definition in section 3.1, the DRL scheme uses two Q functions which consume $2 \cdot 9 \cdot 6 \cdot 9 \cdot 6 \cdot 3 = 17496$ memory cells, versus $9 \cdot 6 \cdot 9 \cdot 6 \cdot 9 = 26244$ of its CRL-9a counterpart; DRL consumes 1/3 less memory. Moreover, we have measured the elapsed

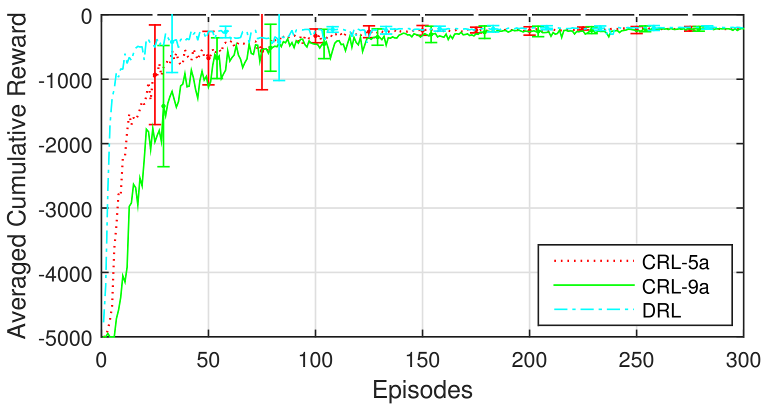


Fig. 3: 3DMC learning evolution plots. Results are averaged across 25 learning runs and error bars show the standard deviation.

time of both learning process along the 25 performed runs, the DRL took 0.62 hours, being 1.56 times faster than CRL-9a, which took 0.97 hours. These times are referential, experiments were performed with an Intel(R)Core(TM)i7-4774CPU@3.40Ghz with 4GB in RAM. Notice than even for this simple problem with only two agents, there is a considerable memory consumption and processing time saving.

Table 2: 3D Mountain Car parameters, final performances, and convergence time.

Approach	**Optimized parameters**	**Final performance**	**Time to Th.**[†]
DRL	$\alpha = 0.2$, $\lambda = 0.95$, $\epsilon = 0.06$	-205.49	71
CRL-5a	$\alpha = 0.25$, $\lambda = 0.95$, $\epsilon = 0.06$	-221.47	146
CRL-9a	$\alpha = 0.2$, $\lambda = 0.95$, $\epsilon = 0.06$	-240.91	195

[†] Time to threshold [12] is defined for this case as the number of episodes to achieve or overcome by first time a performance of -240.91.

4.2 Ball-Pushing

Figure 4 presents learning evolution plots and Table 3 shows the best policy final performances. Notice that learning evolution plots achieves lower performances than final performance presented in Table 3. It is because the performance of learning evolution plots is affected by the poor performance during early learning episodes, meanwhile final performance presented in the table is measured by using the best policy during the whole test. The DRL scheme sped-up learning time and improved the CRL final performance by 15%. If a time to threshold of 37.98% is considered (the best performance achieved by CRL during the learning process shown in Figure 4), it can be said that the DRL scheme learns more than twice as fast as the CRL scheme, achieving this threshold in 441 learning episodes. It can be also noticed from error bars during early episodes, where they do not overlaps between them.

As it was mentioned in section 3.2, number of discretized actions for the linear velocity were optimized obtaining $A^{v_l} = 5$ for the CRL scheme and $A^{v_l} = 7$ for the DRL. Notice that the DRL implementation allows a finer discretization than the CRL. For the CRL case, increasing from 5 to 7 the number of actions of v_l implies increasing the joint action space from 15 to 21 actions, taking into account $A^{a_w} = 3$ (please check Table 1), this implies an exponential increasing in the search space which may increase learning time affecting the final performance. This is one of the interesting properties of decentralized systems, since agents are independents, separate modellings or configurations can be implemented per agent without directly affecting the others.

It is not possible carrying out an equitative comparison between computational consumption of CRL vs. DRL because of the aforementioned 5 vs. 7 discretized actions for v_l. Even so, DRL consumes 1/3 less memory, $625 \cdot 7 + 625 \cdot 3 = 6250$ memory cells, versus $625 \cdot 15 = 9375$ of the CRL implementation (Please see Table 1). On the other hand, the DRL elapsed time was 0.36 hours, almost the same as the 0.34 hours of the CRL scheme.

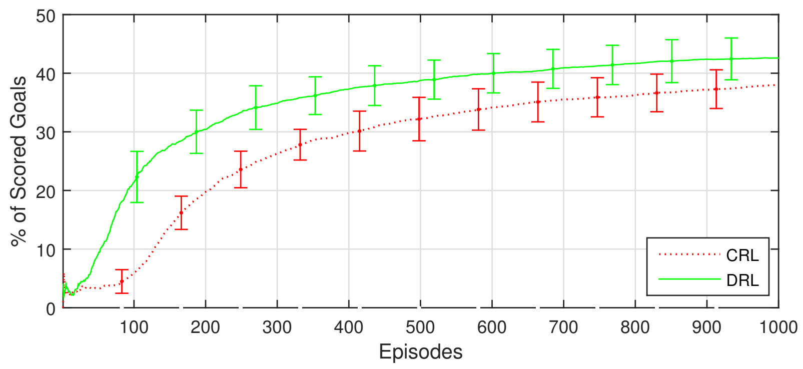


Fig. 4: Ball-pushing learning evolution plots. Results are averaged across 25 learning runs and error bars show the standard deviation.

4.3 Ball-Pushing: physical setup

An experimental setup is implemented in order to test learned policies onto a physical setup, which is shown in Figure 2 at right. The differential drive robot Miabot Pro is used, it is a small cube with a size of $75 \times 75 \times 75\,mm$ and a weight of $0.55kg$. It has two wheels which are driven by two electro motors. The

Table 3: Ball-pushing optimized parameters and best policy final performances of the simulated and real setup

Approach	**Optimized parameters**	**Performance sim. (%)**	**Performance real (%)**
DRL	$\alpha = 0.3$, $\lambda = 0.9$, $\tau_0 = 1$, $dec = 10$	75.28	68.57
CRL	$\alpha = 0.5$, $\lambda = 0.9$, $\tau_0 = 2$, $dec = 7$	62.15	57.14

robot is connected via Bluetooth to a central computer close to the robot soccer platform which is 1.5m x 1m. A web camera above the platform provides position and orientation of the robot, ball, and goal. This central vision system operates at 30 frames per second. The robot position is identified by color segmentation and thresholding. The color patch on the robot make this approach convenient due to its simple nature and ease of implementation. The state observation is processed from the vision system, while speed of the wheels are transmitted through Bluetooth from the computer. These speeds are computed from the Q tables by using a greedy search policy.

The best learned policy obtained between the 25 runs carried out by simulation for the CRL and DRL implementations are directly transferred and tested on the MiaBot Pro robot on the experimental setup. The robot was positioned in seven different positions trying to cover the whole state space, and 10 trials were run from each position. The results from these experiments can be seen in Table 3, where performance is presented in percentage of success to score a goal considering the seventy attempts.

It can be seen from the table 3 that DRL performs on average 11.43% better than CRL. Simulation and physical setup performances are similar which validates simulation experiments and results.

Some experiments for centralized and decentralized RL were recorded and can be seen in [13]. In this video it can be seen that actions are a bit abrupt, it is because of no smoothing or extrapolation of the discrete actions where carried out, policies were transferred directly from Q functions to the physical robot. Also, cases where the mark of the robot or some tracker was lost in the vision system were disregarded. These aspects should be improved for future implementations, however, the porpoise of this work is more focused to compare CRL and DRL approaches, than achieving an optimal performance.

5 Related Work

A multiagent RL application for the multi-wheel control of a mobile robot is presented in [14]; the robots platform is decomposed into driving modules agents that are trained independently, in order to provide energy consumption optimization. In [15], the DRL of the soccer ball-dribbling behavior is accelerated by using transfer knowledge, there, each component of the omnidirectional biped walk (v_x, v_y, v_θ) is learned in parallel with single agents working on a multi-agent task. Similar to [9], where some layered learning strategies are studied and one of them involves the DRL of individual behaviors in the context of soccer robotics.

In [5], definitions of centralized and multiagent learning approach for reinforcement learning are presented. Both learning strategies are tested on a 2-link manipulator, and compared in terms of performance, convergence time, and computational resources. In [3], a distributed RL architecture is presented to learn the inverse kinematics of a 3-link-planar robot and the SCARA robot; experimental results have shown that it is not necessary that the decentralized agents perceive the whole state space in order to learn a good global policy. A multiagent influence RL approach is presented in [16], this uses agents influences to estimate learning error between them; it has been validated with a multi-joined robotic arm.

6 Conclusions

This paper has proposed a decentralized reinforcement learning architecture for implementing behaviors with a mobile robot, where the individual components of the commanded velocity vector are learned in parallel by separate agents working in a multiagent task.

Two validation problems have been modeled and implemented: an extended version of the three dimensional mountain car, and a ball-pushing behavior performed with a differential-drive robot, which has been also tested on a physical setup. A DRL and its CRL counterpart scheme has been implemented for the two validation problems in order to compare and analyze strengths, weaknesses and properties of the DRL proposed framework.

Experimental results have evidenced that with less computational resources, and non direct coordination mechanism, DRL implementations have shown better performances and faster learning times than their CRL counterparts for all the implemented experiments. This empirically demonstrate that benefits of MAS are also applicable to more complex and real world problems like robotic platforms. It opens the door to explore applications with higher dimensional action spaces where a CRL scheme could not be easily implementable, like snake robots, multi-link robotic arms, omni-directional mobile robots, multi-rotor aerial vehicles, etc. Moreover, evaluating MAL algorithms with cooperation and coordination between agents is part of our ongoing schedule.

Acknowledgment

This work was partially funded by FONDECYT under Project Number 1161500. David Leonardo Leottau was funded under grant CONICYT-PCHA/Doctorado Nacional/2013-63130183. The authors would like to thank Technical University of Delft for providing the resources to test the learnt policies on an experimental setup.

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
2. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots **27**(1) (2009) 55–73
3. Martin, J., Lope, H.D.: A distributed reinforcement learning architecture for multi-link robots. In: 4th Int. Conf. on Informatics in Control, Automation and Robotics, ICINCO 2007. Number 3, Angers, Francia (2007) 192–197
4. Busoniu, L., Babuska, R., De-Schutter, B.: A Comprehensive Survey of Multiagent Reinforcement Learning. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **38**(2) (2008) 156–172
5. Busoniu, L., Schutter, B.D., Babuska, R.: Decentralized Reinforcement Learning Control of a Robotic Manipulator. In IEEE, ed.: Ninth International Conference on Control, Automation, Robotics and Vision, ICARCV 2006, 5-8 December 2006, Singapore (2006) 1–6
6. Stone, P., Veloso, M.: Multiagent Systems: A Survey from a Machine Learning Perspective. Autonomous Robotics **8**(3) (2000) 1–57
7. Taylor, M.E., Kuhlmann, G., Stone, P.: Autonomous Transfer for Reinforcement Learning. In: The Autonomous Agents and Multi-Agent Systems Conference (AAMAS). Number May, Estoril, Portugal (2008) 283–290
8. Systems, M.: Miabotpro manual (2016)
9. Leottau, D.L., Ruiz-del solar, J., MacAlpine, P., Stone, P.: A Study of Layered Learning Strategies Applied to Individual Behaviors in Robot Soccer. In Almeida, L., Ji, J., Steinbauer, G., Luke, S., eds.: RoboCup-2015: Robot Soccer World Cup XIX, Lecture Notes in Artificial Intelligence. Volume 2015., Hefei, China, Springer Verlag, Berlin (2016) 290–302
10. Takahashi, Y., Asada, M.: Multi-layered learning system for real robot behavior acquisition. In Kordic, V., ed.: Cutting Edge Robotics. Number July 2005. (2004) 357–375
11. Emery, R., Balch, T.: Behavior-based control of a non-holonomic robot in pushing tasks. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). Volume 3., IEEE (2001) 2381–2388
12. Taylor, M., Stone, P.: Transfer learning for reinforcement learning domains: A survey. The Journal of Machine Learning Research **10** (2009) 1633–1685
13. Vatsyayan, A.: Video: centralized and decentralized reinforcement learning of the ball-pushing behavior. `https://youtu.be/pajMkrf7ldY` (2016)
14. Dziomin, U., Kabysh, A., Golovko, V., Stetter, R.: A multi-agent reinforcement learning approach for the efficient control of mobile robot. In: Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on. Volume 2., IEEE (2013) 867–873
15. Leottau, D.L., Ruiz-del solar, J.: An Accelerated Approach to Decentralized Reinforcement Learning of the Ball-Dribbling Behavior. In: AAAI Workshops, Austin, Texas USA (2015) 23–29
16. Kabysh, A., Golovko, V., Lipnickas, A.: Influence Learning for Multi-Agent System Based on Reinforcement Learning. International Journal of Computing **11**(1) (2012) 39–44