Engineering Applications of Artificial Intelligence 85 (2019) 243–253

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Accelerating decentralized reinforcement learning of complex individual behaviors

David L. Leottau *, Kenzo Lobos-Tsunekawa, Francisco Jaramillo, Javier Ruiz-del-Solar

Department of Electrical Engineering, Advanced Mining Technology Center, Universidad de Chile, Av. Tupper 2007, Santiago, Chile

ARTICLE INFO

Keywords:
Decentralized reinforcement learning
Multi-agent systems
Distributed control
Autonomous robots
Knowledge transfer
Distributed artificial intelligence

ABSTRACT

Many Reinforcement Learning (RL) real-world applications have multi-dimensional action spaces which suffer from the combinatorial explosion of complexity. Then, it may turn infeasible to implement Centralized RL (CRL) systems due to the exponential increasing of dimensionality in both the state space and the action space, and the large number of training trials. In order to address this, this paper proposes to deal with these issues by using Decentralized Reinforcement Learning (DRL) to alleviate the effects of the curse of dimensionality on the action space, and by transferring knowledge to reduce the training episodes so that asymptotic converge can be achieved. Three DRL schemes are compared: DRL with independent learners and no prior-coordination (DRL-Ind); DRL accelerated-coordinated by using the Control Sharing (DRL+CoSh) Knowledge Transfer approach; and a proposed DRL scheme using the CoSh-based variant Nearby Action Sharing to include a measure of the uncertainty into the CoSh procedure (DRL+NeASh). These three schemes are analyzed through an extensive experimental study and validated through two complex real-world problems, namely the inwalk-kicking and the ball-dribbling behaviors, both performed with humanoid biped robots. Obtained results show (empirically): (i) the effectiveness of DRL systems which even without prior-coordination are able to achieve asymptotic convergence throughout indirect coordination; (ii) that by using the proposed knowledge transfer methods, it is possible to reduce the training episodes and to coordinate the DRL process; and (iii) obtained learning times are between 36% and 62% faster than the DRL-Ind schemes in the case studies.

1. Introduction

Reinforcement Learning (RL) is increasingly being used to learn complex behaviors in robotics. Two of the main challenges to be solved for modeling RL systems acting in the real-world are: *(i)* the high dimensionality of the state and action spaces, and *(ii)* the large number of training trials required to learn most of complex behaviors. Many real-world applications have multi-dimensional action spaces (e.g., multiple actuators or effectors). In those cases, RL suffers from the combinatorial explosion of complexity which occurs when a single or centralized RL (CRL) scheme is used. It may turn infeasible to implement CRL systems in terms of computational resources or learning time due to the exponential increasing of dimensionality in both the state space and the action space as in Martín and de Lope Asiaín (2007) and Leottau et al. (2018). Instead, the use of *Decentralized Reinforcement Learning* (DRL) helps to alleviate this problem as it has been empirically evidenced by Buşoniu et al. (2006) and Leottau et al. (2017, 2018). In DRL, a problem is decomposed into several sub-problems, whose resources are managed separately while working toward a common goal, i.e. learning and performing a behavior. In the case of multi-dimensional action spaces, a sub-problem corresponds to control one particular variable. For instance, in mobile robotics, a common high-level motion command is the requested velocity vector (e.g., $[v_x, v_y, v_\theta]$ for an omni-directional robot). Then, if each speed component of this vector is handled individually, a distributed control scheme can be applied.

Most of the stochastic DRL systems implemented with independent learners also present two main drawbacks: non-stationary and non-Markovian issues. Laurent et al. (2011) indicate that these drawbacks could be mitigated by: *(i)* decaying the exploration rate, and *(ii)* using coordinated exploration techniques for shrinking the action space. Both mechanisms can be accomplished by using Knowledge Transfer (KT) approaches, as Knox and Stone (2010) and Bianchi et al. (2014) reported. KT has evidenced to be able to accelerate the training of Multi-Agent RL (MARL) problems (Vrancx et al., 2011; Boutsioukis et al., 2012; Taylor et al., 2013; Bianchi et al., 2014; Hu et al., 2015), alleviating the second aforementioned challenge: the large number of training trials. KT allows exploring in a subset of the action space limited by a source of knowledge (SoK), which has been previously learned or designed. A SoK for a DRL system should contain at least one branch per decentralized learning agent. If those branches are pre-coordinated, the SoK relieves at the same time the coordination problem, which

* Corresponding author.
E-mail address: dleottau@ing.uchile.cl (D.L. Leottau).

https://doi.org/10.1016/j.engappai.2019.06.019
Received 31 December 2017; Received in revised form 14 August 2018; Accepted 26 June 2019
Available online 1 July 2019
0952-1976/

must be solved in order to extend and take advantage of some potential benefits of Multi-Agent Systems (MAS) to DRL systems.

In this work, we tackle the high dimensionality of the state and action spaces, and the large number of training trials by using two mechanisms: *(i)* DRL to alleviate the effects of the curse of dimensionality on the action space, and *(ii)* KT to reduce the training episodes so that asymptotic convergence can be achieved. The SARSA(λ) with radial basis functions (RBFs) is used as basis algorithm to implement a DRL system with no prior-coordination among agents. This is accelerated-coordinated by using a KT approach called Control Sharing (CoSh) introduced by Knox and Stone (2012), which is extended from the single-agent case to the DRL case. In addition, we introduce a CoSh-based variant called Nearby Action Sharing (NeASh), which is able to include a measure of uncertainty in the action sharing process.

Since most of the MARL reported studies do not address or validate their proposed approaches with multi-state, stochastic, and real-world problems (Buşoniu et al., 2008), our preliminary goal is to show (empirically) that the benefits of MAS are also applicable to complex problems by using a DRL scheme. For such purpose, two challenging real-world problems for soccer robotics are modeled and implemented: the *inwalk-kicking* and the *ball-dribbling* behaviors, both performed by using an omni-directional biped robot. In the inwalk-kicking behavior, the robot must learn to push the ball toward a desired target only by using the inertia of its own gait (Lobos-Tsunekawa et al., 2017). In the ball-dribbling problem, the robot must learn to maneuver the ball in a very controlled way while moving towards a desired target (Leottau et al., 2015). In the case of biped robots, the complexity of these tasks is very high, as in each case the controller must take into account the physical interaction among the ball, the robot's feet, the ground, and the robot's gait inertia. Thus, the action is highly dynamic, non-linear, and influenced by several sources of uncertainty.

Three DRL schemes are analyzed and compared for both test problems, the DRL with independent agents and no prior-coordination (DRL-Ind), the DRL accelerated with CoSh (DRL+CoSh), and the DRL accelerated with NeASh (DRL+NeASh). As it will be shown, DRL+CoSh and DRL+NeASh are able to transfer knowledge and accelerate the DRL-Ind. These three schemes are analyzed through an extensive empirical study carried out in a 3D realistic simulator and demonstrated with physical robots. It is worth mentioning that, to the best of our knowledge, we are the first applying an effective strategy for transferring knowledge and coordinating-accelerating DRL systems.

This paper is structured as follows. Relevant background and related work are presented in Section 2. Section 3 describes the proposed DRL and KT schemes. The case studies, their description, experiments and results are presented in Section 4. Finally, conclusions are drawn in Section 5.

2. Background

RL is a family of machine learning techniques in which an agent learns a task by directly interacting with the environment. In the single-agent RL case, the environment of the agent is described by a Markov Decision Process (MDP), this is, a 4-tuple $\langle S, A, T, R\rangle$ where: S is the finite set of environment states, A is the finite set of agent actions, $T : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function, and $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function.

The generalization of the MDP to the MARL case is the stochastic game, defined by the tuple $\langle S, A^1, \ldots, A^M, T, R^1 \ldots R^M\rangle$, where: M is the number of agents; S is the discrete set of environment states; $A^m, m = 1, \ldots, M$, are the discrete sets of actions available to the agents, yielding the joint action set $\mathcal{A} = A^1 \times \cdots \times A^M$; $T : S \times \mathcal{A} \times S \rightarrow [0, 1]$ is the state transition probability function, such that, $\forall s \in S, \forall a \in \mathcal{A}, \sum_{s' \in S} T(s, a, s') = 1$; and $R^m : S \times \mathcal{A} \times S \rightarrow \mathbb{R}, m = 1, \ldots, M$, are the reward functions of the agents (Buşoniu et al., 2008; Laurent et al., 2011).

2.1. Decentralized reinforcement learning

Under the presence of multi-dimensional action spaces, RL solutions can be called CRL systems if each action subspace is discretized, combined, and computed as a single set of actions. In CRL, the number of possible actions grows exponentially with the dimensionality of the action space. Thus, a combinatorial explosion of both state and action spaces occurs, making hard exploring sufficiently the whole action–state space which causes a slow convergence and an exponential increasing of the execution time and memory consumption as Leottau et al. (2018) and Lobos-Tsunekawa et al. (2017) evidenced. These drawbacks can be overcome by addressing them from a decentralized perspective.

A DRL system uses a single-entity which has several branches (e.g., a single robot with multiple actuators). In DRL, a problem is split into several sub-problems, and their individual information and resources are then managed in parallel as a collection of multiple separate learning agents, which are part of a single-entity. Under multi-dimensional action spaces, each separate agent acts in a different action space dimension, this allowing the design of independent state models, reward functions, and learning agents for each action dimension (Leottau et al., 2018).

DRL helps to alleviate the high dimensionality of the state and action spaces, and the large number of training trials. The DRL's multi-agent nature grants several potential advantages if the problem is approached with decentralized learners and the coordination issue is solved (Leottau et al., 2018):

- Since all the separate agents in a DRL system can operate in parallel, acting on their individual action spaces, then the learning speed is higher with respect to a centralized agent which searches an exponentially larger action space $A = A^1 \times \cdots \times A^M$.
- If not all the state information is relevant for a particular agent, its state space can be reduced.
- Different algorithms, modelings or configurations could be used independently by each separate agent.
- Memory and computing/processing time requirements are reduced (Lobos-Tsunekawa et al., 2017 shows that even for relatively low-dimensional problems, the computational advantages of CRL are significant, as it produces a practical speedup of x69 and a memory consumption reduction of x85)
- Parallel or distributed computing implementations are allowed.

2.1.1. Challenges in DRL

DRL systems also have several challenges which must be solved efficiently in order to take advantage of the MAS benefits already mentioned. Agents have to coordinate their individual behaviors toward a coherent-desired joint behavior. This is not a trivial issue since those single behaviors are correlated, and each individual decision modifies the joint environment.

According to Claus and Boutilier (1998), two fundamental classes of agents in MAS can be defined: *(i)* joint-action learners, and *(ii)* independent learners (ILs). Joint-action learners are able to observe the other agents' actions and rewards; these learners are easily generalized from standard single-agent RL algorithms as the process stays Markovian. On the other hand, ILs do not observe the rewards and actions of the other learners, and they interact with the environment as if no other agents exist (Laurent et al., 2011).

Most multi-agent (MA) stochastic problems violate the Markov property and are non-stationary. A process is said non-stationary if its transition probabilities change with the time. A non-stationary process can be Markovian if the evolution of its transition and reward functions depends only on the time step, and not on the history of actions and states (Laurent et al., 2011). For ILs, which is the focus of the present paper, it is expected that the individual policies change as the learning progresses. A past action of an IL agent has influenced

the past evolution of its learning process and the future evolution of its environment; thus, the learned behavior of other agents may also have changed. Therefore, the environment is non-stationary and non-Markovian, causing convergence issues.

ILs equipped with single-agent algorithms are more likely to converge in the case of low-coupled distributed systems, because the effects of the non-stationarity of agents are less observable. However, achieving convergence for these low coupled systems requires to decay the exploration rate of new actions as the learning process goes along in order to avoid too much concurrent exploration. Another strategy for mitigating ILs convergence issues is to use coordinated exploration techniques; by excluding one or more actions from each independent action space in a coordinated way, the action selection mechanism explores in a shrinking and joint action space. Notice that both mentioned strategies reduce the exploration of new actions, the agents evolve slower, and the non-Markovian effects are reduced as Laurent et al. (2011) mention.

2.2. Knowledge transfer and DRL

KT is used to accelerate the rate at which one or more target tasks are learned from one or more sources of knowledge. Two reasonable goals of KT are: *(i)* to effectively reuse past knowledge in a novel task, and *(ii)* to reduce the overall time required to learn a complex task. In the context of DRL-Ind, we will consider a third goal: to address the coordination problem.

KT has been widely studied and applied to accelerate single-agent RL (Taylor and Stone, 2009). To a lesser extent, KT has been used for MARL systems as well, and not only to accelerate but also to outperform and address large-scale problems as Vrancx et al. (2011) and Bianchi et al. (2014). In order to deal with non-stationary and non-Markovian issues in most of the stochastic IL problems, it is suggested to decay the exploration rate and use coordinated exploration techniques for excluding some actions (Laurent et al., 2011). Both considerations can be accomplished by using KT. In fact, decayed exploration and transfer rates are commonly considered parameters in some KT approaches like (Knox and Stone, 2010; Bianchi et al., 2014). KT also allows shrinking the action space based on a prior-coordinated source task, this simultaneously helping the coordination problem as Vrancx et al. (2011) report for the MARL case.

Different cases of KT applied to DRL tasks can be identified: same tasks (source and target) and same problem-spaces; same tasks and different problem-spaces; different tasks and same problem-spaces; and different tasks and different problem-spaces. The problem-spaces refer to the source and target state variables and actions. In addition, DRL systems consider the case of homogeneous and heterogeneous agents, in which homogeneous agents have identical problem-spaces and goals.

For this work, we consider the following cases:

- Heterogeneous agents, in order to allow designing an individual goal for each independent learner.
- Different source and target tasks, in order to allow the use of layered learning (Leottau et al., 2017) and easy mission approaches (Takahashi and Asada, 2005).
- The same source and target problem-space, in order to avoid source-task-selection or task-mapping (Taylor and Stone, 2009), which may slow-down and complicate the procedure.

2.3. Related work

Since this article considers two main methods, namely independent DRL and KT applied to MARL, this section presents some related work about these two approaches.

2.3.1. Distributed and decentralized RL

Leottau et al. (2018) introduce a MA methodology for DRL of individual behaviors in problems where multi-dimensional action spaces are involved. This modeling-design methodology consists of five stages: *(i)* determining if the problem is decentralizable; *(ii)* identifying common and individual goals; *(iii)* defining the reward functions; *(iv)* determining if the problem is fully decentralizable; and *(iii)* completing RL single modelings. This work also reports an experimental study which evidences the benefits of DRL implementations over their CRL counterparts. Some of those results can be seen in Fig. 1, which compares CRL vs. DRL learning evolution plots of three different problems: the well-known 3-Dimensional mountain car, the ball-pushing behavior performed with a differential drive robot, and the ball-dribbling tested with a simplified biped robot.

As previous work to this paper, the DRL of the soccer Ball-Dribbling behavior is accelerated by using knowledge transfer in Leottau and Ruiz-Del-Solar (2015), where each component of the omni-directional biped walk (v_x, v_y, v_θ) is learned in parallel with single-agents working on a MA task. This learning approach for the omni-directional velocity vector is also reported by Lobos-Tsunekawa et al. (2017), where the inwalk-kicking problem is proposed and tested in biped robots, using finite support basis functions for that purpose. Similarly, a MARL application for the multi-wheel control of a mobile robot is presented by Dziomin et al. (2013). There, the robot's platform is separated into driving module agents that are trained independently, in order to provide energy consumption optimization. It is worth mentioning that to the best of our knowledge, only few works have reported applications in which the commanded velocity vector of a robot is controlled by a DRL system, such as we are proposing in this article.

Some other works have been reported, which in contrast to our humanoid biped robot applications, apply DRL to multi-link robots and arms. Buşoniu et al. (2006) compare centralized and decentralized RL approaches for the case of controlling a 2-link manipulator, in which both learning strategies were tested and compared in terms of performance, convergence time and computational resources. Martín and de Lope Asiaín (2007) present a distributed RL architecture for generating a real-time trajectory of both a 3-link-planar robot and the SCARA robot; experimental results showed that it is not necessary for decentralized agents to perceive the whole state space in order to learn a good global policy. Troost et al. (2008) use a MA approach in which each output is controlled by an independent $Q(\lambda)$-learning agent. Both simulated robotic systems tested showed an almost identical performance and learning time between the single-agent and MA approaches, while the latter requires less memory and computation time. Some of these experiments and results were extended and presented by Schuitema (2012). A multi-agent influenced RL approach is presented by Kabysh et al. (2012), which uses agent's influences to estimate learning error among all the agents. This method has been validated with a multi-joint robotic arm.

2.3.2. KT applied to MARL

To the best of our knowledge, the present paper is the first work reporting a DRL implementation accelerated-coordinated by using KT. Thus, this section briefly overviews relevant and similar works but in the context of KT applied to MARL. Hu et al. (2015) present the idea of equilibrium transfer based MARL, which outperforms and accelerates considerably its non-transfer counterpart, and scales significantly better than algorithms without equilibrium transfer when the state/action space grow and the number of agents increases. Compared to the KT methods used in our work, this method only has been validated on discrete domains, and several non-trivial considerations must be taken into account for implementing effectively the transfer approach. However this method seems an interesting alternative for future DRL implementations. Bianchi et al. (2014) present the heuristically accelerated MARL (HAMRL) algorithms, as a general framework for including heuristic functions to influence the action choice of the agents. This

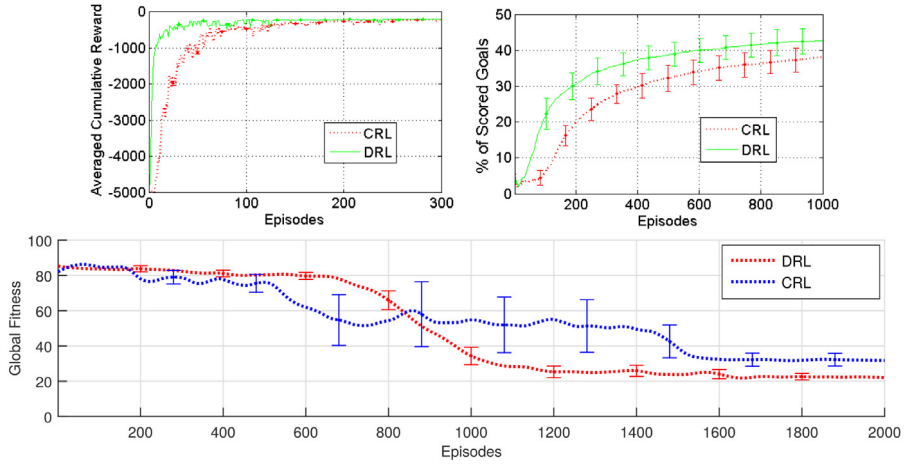

Fig. 1. CRL vs. DRL learning evolution comparison: 3D-Mountain-car (left-top), ball-pushing (right-top), and ball-dribbling (bottom).

family of algorithms must be modified and adapted depending on the source and target MARL algorithm to be used; however, it is also an interesting approach for future implementations, since CoSh can be seen as a simplified version of HAMRL. Vrancx et al. (2011) apply transfer learning to the Coordinating Q-learning framework, which uses a statistical test to sample the immediate rewards received by the agent. This approach shows interesting results and it is worth considering it for future and more sophisticated DRL implementations. By contrast to the KT methods used in our work, this algorithm uses previously identified problem states as samples to train a rule based classifier, which makes somehow complex a fast implementation. Taylor et al. (2013) propose a parallel transfer learning for MAS which, unlike to the approach here-proposed, is able to learn simultaneously the source and target tasks, and share their current experience based on a set of rules and considerations somehow focused on the particular case studied, a smart grid. Boutsioukis et al. (2012) apply transfer learning in MARL domains, showing that the transfer method reduces the learning time and increases the asymptotic performance. Similar to the KT-DRL approaches used in our work, this method biases the initial action value function, but it is only validated in a discrete, deterministic and 2-agents competitive domain.

3. Proposed DRL schemes

3.1. Independent DRL

The DRL-Ind scheme aims to apply single-agent RL methods to the MARL task, and does not consider any kind of cooperation or coordination among agents; there is neither adaptation to the other agents nor estimated models of their policies, nor special action-selection mechanisms (e.g., communication among agents, prior knowledge). The computational complexity of this DRL scheme is the same as that for a single-agent RL (e.g., a Q-Learner).

Although the non-stationarity of the MARL problem invalidates most of the single-agent RL theoretical guarantees, this approach has been implemented in several multi-robot systems. An empirical study about DRL-Ind effectiveness can be found in Leottau et al. (2018).

Algorithm 1 depicts an episodic multi-agent SARSA algorithm (Sutton and Barto, 1998) for continuous states with RBF approximation (Papierok et al., 2008), built following the DRL-Ind scheme. There, a learning system with an M-dimensional action space is modeled with M single SARSA learners acting in parallel. Every IL has individual Q-functions, action spaces, action selection mechanisms, and state vectors. Taking advantage of parallel computation of MAS, it is possible to update every Q-table by using M independent threads (Lines 1.20 to 1). A decayed and synchronized exploration rate is proposed, in order to avoid too much concurrent exploration and reduce the non-Markovian effects as was suggested in Section 2.1.1. In this way, each agent should find the best response to the behavior of the others. Thus, an ϵ-greedy action selection mechanism is implemented, which is exponentially decayed by using the dec factor as seen in Line 1.31; $episode$ is the current episode index and $maxEpisodes$ is the total number of trained episodes per run.

Synchronizing the exploration–exploitation mechanism among all the agents is a variant that Algorithm 1 offers. It is possible just by declaring a unique random number for all the agents as in Line 1.16, instead of an individual random scalar per agent as in Line 1.18. Also note that the RL parameters could be defined separately per agent (e.g., $\alpha^m, \gamma^m, \epsilon^m$), which is one of the DRL properties pointed out in Section 2.1. In Algorithm 1, those parameters are unified just for the sake of simplicity.

3.2. Proposed KT-based DRL

This work proposes to use KT to help DRL-Ind in coordinating the exploration during the early episodes. Under this approach, a subset of actions taken from a prior-coordinated SoK is used for guiding the agents to avoid unknown actions, while they evolve leniently and the non-Markovian effects are reduced. The SoK acts as an initial value function, and thereby endows the agent with an initial policy (Konidaris et al., 2012). If both decayed exploration rate and decayed probability of KT are used, the subset of actions from the SoK is progressively increased or modified over time, while concurrent exploration is reduced. In this way, each agent finds easily the best response to the behavior of the others.

Several requirements are taking into account in order to choose the KT strategy used in this work. In general, we consider methods that are able to:

1. Transfer toward any RL algorithm that uses an action value function, such as the works reported by Taylor and Stone (2007), Knox and Stone (2010), Mataric (1994), Boutsioukis et al. (2012) and Bianchi et al. (2014).
2. Transfer from different SoK types such as standard controllers (e.g. linear or fuzzy controllers), hand-craft behaviors, and RL policies such as the works reported in Fernández et al. (2010), Knox and Stone (2010) and Bianchi et al. (2014).
3. Scale-up when the state/action space grow and the number of agents increases, but without extra memory consumption, such as Vrancx et al. (2011), Knox and Stone (2010) and Bianchi et al. (2014)
4. Avoid to build empirical or on-line models of the other agents strategies such as Knox and Stone (2010) and Bianchi et al. (2014).

Algorithm 1 DRL Independent: MA-SARSA with RBF approximation and ϵ-greedy exploration

Parameters:
M ▷ Number of decentralized learning agents
$\epsilon \leftarrow \epsilon_0$ ▷ Initial exploration probability $\in (0, 1]$, typically $\epsilon_0 = 1$
dec ▷ Decay exploration rate: $dec = 0$ no decay; more decay if dec is higher
α ▷ Learning rate $\in (0, 1]$
γ ▷ Discount factor $\in (0, 1]$
Φ^m ▷ Size of the feature vector ϕ^m of $agent_m$, where $m = [1, \cdots, M]$
Inputs:
$S^1, \cdots, S^M$ ▷ State space of each agent
$A^1, \cdots, A^M$ ▷ Action space of each agent
Initialize $\vec{\theta}^m$ arbitrarily for each agent $m = 1, \cdots, M$
procedure FOR EACH EPISODE:
  for all agent $m \in M$ **do**
    $a^m, s^m \leftarrow$ Initialize state and action
  end for
  repeat for each step of episode:
    if Synchronized exploration **then**
      $urnd \leftarrow$ a uniform random variable $\in [0, 1]$
    else
      $[urnd^1, \cdots, urnd^M] \leftarrow$ a uniform random vector $\in [0, 1]$
    end if
    for all agent $m \in M$ **do**
      Take action $a = a^m$ from current state $s = s^m$
      Observe reward r^m, and next state $s' = s'^m$
      if $urnd^m > \epsilon$ **then**
        for all action $i \in A^m(s')$ **do**
          $Q_i \leftarrow \sum_{j=1}^{\Phi^m} \theta_i^m(j) \cdot \phi_{s'}^m(j)$
        end for
        $a' \leftarrow \text{argmax}_i Q_i$
      else
        $a' \leftarrow$ a random action $\in A^m(s')$
      end if
      $\epsilon = \epsilon_0 \cdot \exp(dec \cdot episode/maxEpisodes)$
      $Qas = \sum_{j=1}^{\Phi^m} \theta_a^m(j) \cdot \phi_s^m(j)$
      $Qas' = \sum_{j=1}^{\Phi^m} \theta_{a'}^m(j) \cdot \phi_{s'}^m(j)$
      $\delta \leftarrow r^m + \gamma \cdot Qas' - Qas$
      $\theta_a^m \leftarrow \theta_a^m + \alpha \cdot \delta \cdot \phi_s^m$
      $s^m \leftarrow s', a^m \leftarrow a'$
    end for
  until Terminal condition
end procedure

5. Avoid to consider other agents' actions nor action choice negotiation mechanisms such as Knox and Stone (2010) and Hu et al. (2015).

Note that the CoSh approach (Knox and Stone, 2010) accomplishes all the listed considerations. In addition, CoSh also supports the KT case already indicated in Section 2.2: heterogeneous agents, different source and target tasks, and the same source and target problem-space.

3.2.1. Control sharing (CoSh)

Introduced by Knox and Stone (2010), CoSh acts only during action-selection, without affecting the updates of the Action-Value functions. This method effectively either lets the RL agent to choose its action or takes a_{src}, the action shared from the SoK. If an action is shared, the RL agent observes and updates it as if it were making the choice. The action a is chosen by SoK or *source-policy* ($\pi_{src}(s) = a_{src}$) with probability β as

$$P(a = a_{src}) = min(\beta, 1), \tag{1}$$

otherwise action a is chosen using a base RL agent's action-selection mechanism. β is decayed periodically by a predefined factor.

CoSh was originally proposed for the single-agent RL case, but it can be easily extended to the DRL case if a SoK is available to each separate agent and if decayed and synchronized exploration rates are implemented as in Algorithm 1, as well as with the transfer probabilities are described as below. Note that CoSh is able to accomplish all the requirements previously mentioned precisely because its extreme simplicity. This is an advantage in order to quick implementations because only the decayed probability β must be set. However, in the next subsection, we are proposing the Nearby Action Sharing variant for continuous action spaces, which also fulfills those requirements and additionally is able to include a measure of uncertainty to the transferred action for noisy SoKs.

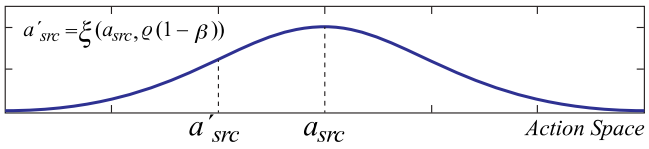

Fig. 2. Normal random function proposed to NeASh approach.

3.2.2. Nearby action sharing (NeASh)

Same as CoSh, NeASh also acts only during action-selection, transferring knowledge from continuous action spaces, when no information different to the suggested action in an observed state is available from the SoK. NeASh has applicability in cases where the sources of knowledge are standard controllers, hand-coded behaviors, rule inference systems, among other similar sources. NeASh is based on CoSh, but it takes advantage of continuous action spaces to compensate the lack of information or uncertainty about the quality of the source actions. It assumes that a measure of the quality of a state–action pair is related to its distance to the action a_{src} suggested by the SoK (e.g., a source policy $\pi_{src}(s)$). In this way, a normal distribution along the universe of discourse centered in a_{src} is considered (see Fig. 2), and the resulting nearby action to share is $a'_{src} = \xi(\mu, \sigma)$, in which $\xi(\mu, \sigma)$ is a normally distributed random generator with mean $\mu = a_{src}$ and standard deviation $\sigma = \varrho(1 - \beta)$. Algorithm 2 depicts the procedure for a DRL system accelerated-coordinated by using NeASh, which has M single-agents.

As in the CoSh case, the action is chosen by *source-policy* with probability β. Typically, the initial value of β is 1. β is decayed periodically as well as the standard deviation of ξ, which means that, at the beginning of the learning process, NeASh works similarly to CoSh in Eq. (1). However, while the learning process goes along, the probability of choosing an action a'_{src} increasingly goes away from the action a_{src} as $\varrho(1-\beta)$. Actually, NeASh turns into CoSh for the particular case of $\varrho = 0$.

If no KT is selected during a step, as in the case of line 2.22, then NeASh offers the option of using its own action-selection mechanism or just using a regular approach (e.g., ϵ-greedy) as depicted in line 2.28. The NeASh action-selection mechanism works similar to Softmax (Sutton and Barto, 1998), but taking advantage of the continuous action spaces, and uses a normal distribution instead of a Boltzmann one. In a very similar way to obtaining a'_{src}, the chosen action from the target policy a'_{tgt} is obtained by using the best action from the current target policy (e.g., $\max Q^m(s^m)$), but with $\sigma = \varrho \cdot \beta$, and taking a nearby target action as in line 2.26.

A synchronized transfer/exploration version of NeASh can be implemented by using a unique random number as in Lines 2.8–2.9, instead of M different random numbers for synchronizing transfer/exploration such as in Lines 2.11–2.13. Note that if normal distributions are used, it is necessary to bound a'_{src} and a'_{tgt} into the action space with module or clip functions. This issue can be solved by using other finite support kernels such as triangular, cosine or Epanechnikov (Lobos-Tsunekawa et al., 2017).

Algorithm 2 DRL+NeASh: KT and action selection mechanism

Parameters:
M ▷ Number of decentralized learning agents
ϱ^m ▷ Scale factor for the continuous action space of $agent_m$, where $m = [1, \cdots, M]$, $\varrho \geq 0$
β ▷ Probability of choosing the action from π_{src}. β is periodically decayed $\in [0, 1]$
Inputs:
$S^1, \cdots, S^M$ ▷ State space of each agent
$A^1, \cdots, A^M$ ▷ Action space of each agent
repeat for each step:
  if Synchronized exploration and transfer **then**
    $urnd \leftarrow$ a uniform random variable $\in [0, 1]$
    $\xi_T, \xi_E \leftarrow$ normal random variables $(\mu = 0, \sigma = 1)$ for transferring and exploration procedures respectively
  else
    $[nrnd^1, \cdots, urnd^M] \leftarrow$ a uniform random vector $\in [0, 1]$
    $[\xi_T^1, \cdots, \xi_T^M] \leftarrow$ a normal random vector $(\mu = 0, \sigma = 1)$
    $[\xi_E^1, \cdots, \xi_E^M] \leftarrow$ a normal random vector $(\mu = 0, \sigma = 1)$
  end if
  for all agent $m \in M$ **do**
    $s^m \leftarrow$ Get state of $agent_m$
    if $urnd^m < \beta$ **then**
      $a^m_{src} \leftarrow \pi_{src}(s^m)$ Get the action from the source-policy for $agent_m$
      μ += a^m_{src}
      σ *= $\varrho^m(1 - \beta)$
      $a^m \leftarrow a'^m_{src} = \xi^m_T(\mu, \sigma)$
    else if NeASh action-selection mechanism is used **then**
      $a^m_{tgt} \leftarrow$ Get the best action from the current target policy of RL $agent_m$
      μ += a^m_{tgt}
      σ *= $\varrho^m \cdot \beta$
      $a^m \leftarrow a'^m_{tgt} = \xi^m_E(\mu, \sigma)$
    else
      $a^m \leftarrow$ Set action from the current RL action selection mechanism of $agent_m$
    end if
  end for
until Terminal condition

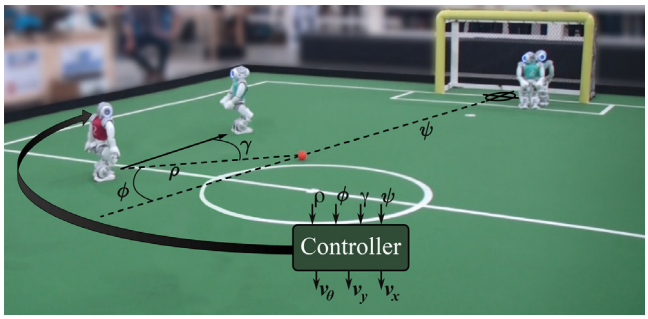

Fig. 3. Geometric state variables and control actions for the ball-pushing based behaviors, performed by the NAO robot using a magenta jersey in a real RoboCup game. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Experiment's acronyms and their optimized parameters.

Acronym	Algorithm's parameters
Inwalk-kicking	
2D simulator	
DRL-Ind	$\epsilon_0 = 1,\ dec = 30,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+NeASh-SrcHQ	$\varrho = 25,\ \beta = 1,\ dec = 29,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+NeASh-SrcLQ	$\varrho = 16,\ \beta = 1,\ dec = 12,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+CoSh-SrcHQ	$\epsilon_0 = 1,\ \beta = 1,\ dec = 19,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+CoSh-SrcLQ	$\epsilon_0 = 1,\ \beta = 1,\ dec = 13,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
3D realistic simulator	
DRL-Ind	$\epsilon_0 = 1,\ dec = 30,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+NeASh	$\varrho = 10,\ \beta = 1,\ dec = 15,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+CoSh	$\epsilon_0 = 1,\ \beta = 1,\ dec = 15,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
Ball-dribbling	
3D realistic simulator	
DRL-Ind	$\epsilon_0 = 1,\ dec = 20,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+NeASh	$\varrho = 5,\ \beta = 0.5,\ dec = 10,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
DRL+CoSh	$\epsilon_0 = 1,\ \beta = 1,\ dec = 30,\ \alpha = 0.2,\ \lambda = 0.9,\ \gamma = 0.99$
RL-FLC	Final performance taken from Leottau et al. (2016)

4. Experimental validation

In order to validate MAS benefits and properties of the DRL proposed approaches, two complex and real-world problems from soccer robotics have been selected: the *inwalk-kicking* and the *ball-dribbling*. Both behaviors are performed with humanoid biped robots, which results very challenging because their modeling must take into account the physical interaction between the ball, the ground, the robot's feet, and the robot's gait inertia. Thus, the action is highly dynamic, non-linear, and influenced by several sources of uncertainty. Moreover, these two behaviors are actually used by the UChile Robotics Team (Yáñez et al., 2016) in the RoboCup (Veloso and Stone, 2012) Standard Platform League (SPL) soccer competition, in which the team has been regular semifinalist in the recent world competitions.

The inwalk-kicking and ball-dribbling are part of the *inwalk-ball-pushing based behaviors* proposed by Leottau and Ruiz-Del-Solar (2015). Similar to Leottau et al. (2015), the description of both problems use the following variables: $[v_x, v_y, v_\theta]$, the velocity vector; γ, the robot-ball angle; ρ, the robot-ball distance; ψ, the ball-target distance; and ϕ, the robot-ball-target complementary angle. These variables are shown in Fig. 3, where the desired target $\oplus$ is the opponent's goal, and a robot's egocentric reference system is indicated with the x axis pointing forwards. Fig. 3 also shows the RoboCup SPL soccer environment where the NAO humanoid robot is used (Gouaillier et al., 2009).

A description of each problem as well as the implementation and modeling details are presented in the next sub-sections. The experimental results are then discussed, for which we use the following terminology: DRL-Ind is an independent learners scheme implemented without any kind of MA coordination; DRL+CoSh and DRL+NeASh are DRL schemes accelerated using CoSh and NeASh transfer knowledge approaches, respectively; RL-FLC is an implementation reported by Leottau et al. (2015, 2016), which combines a Fuzzy Logic Controller (FLC) and an RL single agent. For the kicking problem, some extra experiment are carried out by using different sources of knowledge for CoSh and NeASh transfer methods, namely SrcHQ and SrcLQ, a high and a low quality sources, respectively. This is explained in Section 4.1.2. All the acronyms of the implemented methods and problems are listed in Table 1.

4.1. Inwalk kicking

In the inwalk-kicking behavior, a robot attempts to shoot and score a goal by performing an *inwalk-ball-pushing* (Lobos-Tsunekawa et al., 2017). A general version of this behavior was originally introduced and implemented by Röfer et al. (2011), in which the gait phases are modified to create kick motions. Our proposed implementation consists of inwalk kicks using only the inertia of the gait, without any specially designed kick motion. The robot just push the ball as hard as possible while it is walking toward the ball, as Lobos-Tsunekawa et al. (2017) propose.

4.1.1. Decentralized modeling

Since the velocity vector of the biped-robot walking-engine is $[v_x, v_y, v_\theta]$, it is possible to decentralize this 3-Dimensional action space by using three separate agents, namely $Agent_x$, $Agent_y$, and $Agent_\theta$. Our

Table 2
Description of state and action spaces for the DRL modeling of the inwalk-kicking problem.

Joint state space: $S = [\rho, \gamma, \phi, \psi]^T$			
State variable	Min.	Max.	N. Cores
ρ	0 mm	800 mm	15
γ	−70°	70°	11
ϕ	−90°	90°	13
Action space: $A = [v_x, v_y, v_\theta]$			
Agent	Min.	Max.	N. Actions
v_x	0 mm/s	120 mm/s	16
v_y	−70 mm/s	70 mm/s	15
v_θ	−30 °/s	30 °/s	17

expected common goal is to walk fast toward the ball, and pushing it aligned and hard enough for scoring a goal. That means: to maximize v_x and minimize $\rho, \gamma, \phi, v_y, v_\theta$ before pushing the ball; and to minimize ϕ, ψ once the ball is shoot. So, the proposed control signals are $[v_x, v_y, v_\theta]$, and the proposed common reward is:

$$r = \begin{cases} K \cdot \exp\left(-\psi_{error}/\psi_0\right) \cdot \exp\left(-\alpha_{error}/\alpha_0\right), & \text{if } ball \text{ is } pushed, \\ -\left(\rho/\rho_{max} + |\phi|/\phi_{max} + |\gamma|/\gamma_{max}\right), & \text{otherwise,} \end{cases} \quad (2)$$

where $[\rho_{max}, \gamma_{max}, \phi_{max}] = [2000\text{ mm}, 90°, 90°]$, ψ_{error} is the distance that the ball still needs to travel to reach the target in its current trajectory, α_{error} is the angle deviation of the ball's trajectory from a straight line to the target, and the parameters K, ψ_0 and α_0 allow the design of rewards with more focus on kick strength or precision. The complete proposed modeling for learning the 3-Dimensional velocity vector from the joint observed state is detailed in Table 2.

4.1.2. Experimental setup

The inwalk-kicking RL procedure is carried out episodically. After a reset, the ball is set on a fixed position, 1.5 m in front of the opposite goal as shown in Fig. 4 (left). The robot is set on a random position, 1m around the ball and always facing it. This random initialization is designed so the learning agent can successfully learn many operation points, and thus achieve a general kick behavior. The episode's termination criterion is given by the following conditions: episode timeout of 200 s, the robot or the ball leaves the field, or the ball is pushed by the robot.

A SARSA(λ) RL algorithm with RBF approximation is implemented for these experiments. DRL-Ind and DRL+CoSh use ϵ-greedy decayed as

$$\epsilon = \epsilon_0 \cdot \exp(decepisode/maxEpisodes) \quad (3)$$

where dec is a decayed factor, *episode* is the current episode index, and $maxEpisodes = 2000$ trained episodes per run. DRL+CoSh also decay β in the same way. DRL+NeASh uses the same decay function, but applied for annealing β and managing the knowledge transfer and the NeASh action-selection mechanism depicted in Algorithm 2.

The percentage of scored goals across the trained episodes is considered as the performance index

$$ScoredGoalRate(\%) = scoredGoals/EpisodeWindow \quad (4)$$

where $scoredGoals$ are the number of scored goals during $EpisodeWindow = 200$ episodes, a window of 10% of the 2000 total trained episodes.

Two kinds of experiments are carried out: *(i)* an extensive experimental procedure carried out on a 2D simulator in which basic kinematics models are computed faster, allowing parameter optimizations and running several trials for more statistical significance; and *(ii)* experiments carried out in the SimRobot 3D simulator released by Röfer et al. (2011), which is very realistic but computationally expensive for the Intel(R)Core(TM)i7-4774CPU@3.40 GHz available on our lab (a run of 2000 episodes may take up to 12 h). The experimental procedure for both experiments is described below:

2D Simulator Experiments:

- Five different schemes are tested: DRL-Ind, DRL+CoSh-SrcHQ, DRL+CoSh-SrcLQ, DRL+NeASh-SrcHQ, and DRL+NeASh-SrcLQ. Since NeASh and CoSh approaches require a source for transferring knowledge, a linear controller of the form:

$$\begin{bmatrix} v_x \\ v_y \\ v_\theta \end{bmatrix} = KX = \begin{bmatrix} k_{x\rho} & \cdots & k_{x\phi} \\ \vdots & \ddots & \vdots \\ \cdot & \cdots & k_{\theta\phi} \end{bmatrix} \begin{bmatrix} \rho \\ \gamma \\ \phi \end{bmatrix} \quad (5)$$

  with two configurations: SrcHQ, a high quality source in which matrix $\boldsymbol{K}$ in Eq. (5) was tuned for the best performance achievable by these linear controllers (around 20% on average); SrcLQ, a low quality source in which $\boldsymbol{K}$ was tuned only for achieving the ball without kicking it. These two different sources of knowledge are tested in order to analyze their impact in the final performance and learning time.
- The decay factor (dec) and the action space scale factor (ϱ) parameters were optimized for each of these five implementations by using the custom hill-climbing algorithm presented by Leottau et al. (2018). This is an important step in order to guarantee that every scheme tested uses the best parameter settings. In this way, our comparisons and evaluations are carried out based on the best performance potentially achievable by each method, according to our optimization results. All the parameters are detailed in Table 1.
- The best set of parameters of each implemented scheme is evaluated 25 times and learning evolution plots are averaged. Final performances are measured as well as the number of episodes required to achieve a given performance, which is called *time to threshold*.

3D Realistic Simulator Experiments:

- In order to validate the 2D simulator experiments, three implementations are tested: DRL-Ind, DRL+CoSh-SrcHQ, and DRL+NeASh-SrcHQ. The averaged performance of SrcHQ for this case is around 15%.
- The decay factor (dec) and the action space scale factor (ϱ) parameters were also optimized for each of the three said implementations. Due to computing time limitations common in complex 3D simulators, a grid search of the whole parameter space was used in order to find the best sets of parameters.
- The best set of parameters of each implemented scheme is evaluated 10 times and learning evolution plots are averaged.

4.1.3. Results and analysis

Fig. 5 shows learning plots for 2D simulator experiments. Table 3 presents their final performances and learning times for achieving a time to threshold of 40%. DRL+NeASh schemes show the best final performance and learning times (around 14% better and 43% faster than DRL-Ind) followed by DRL+CoSh schemes (around 10% better and 36% faster than DRL-Ind). DRL-Ind shows the lowest performance and slower learning times, which is expected taking into account the lack of prior-coordination, unlike NeASh and CoSh approaches which use linear controllers as SoK.

Two different qualities of the SoK were tested for comparing NeASh and CoSh performances. Both transfer methods outperform the DRL-Ind even when a low quality source is used. However, note from Fig. 5 that the DRL+CoSh-SrcLQ learning plot shows lower performance during most of the learning procedure, evidencing that the quality of the SoK affects CoSh more than NeASh. From Table 3, DRL+NeASh-SrcHQ is about 21% faster than DRL+CoSh-SrcLQ, and 23% faster than DRL-Ind.

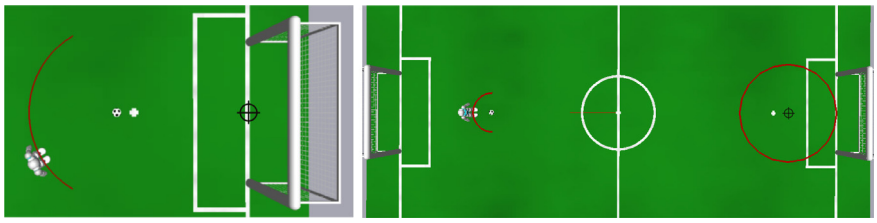

Fig. 4. The learning setup environment of the inwalk-kicking problem (left), and the ball-dribbling problem (right).

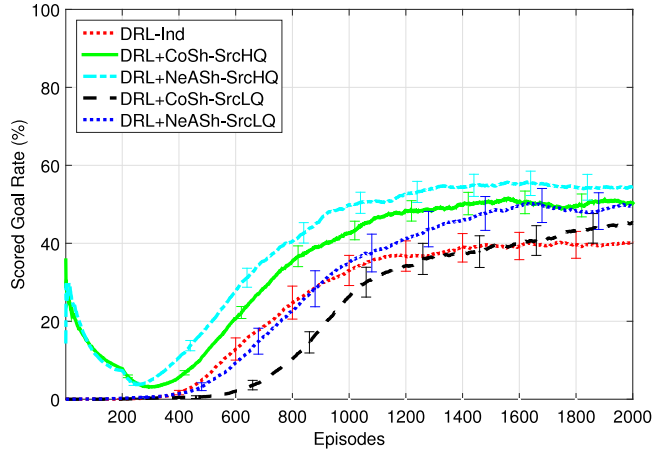

Fig. 5. Inwalk-kicking learning evolution plots with two different sources of knowledge. Results are averaged across 25 learning runs and error bars show the standard errors.

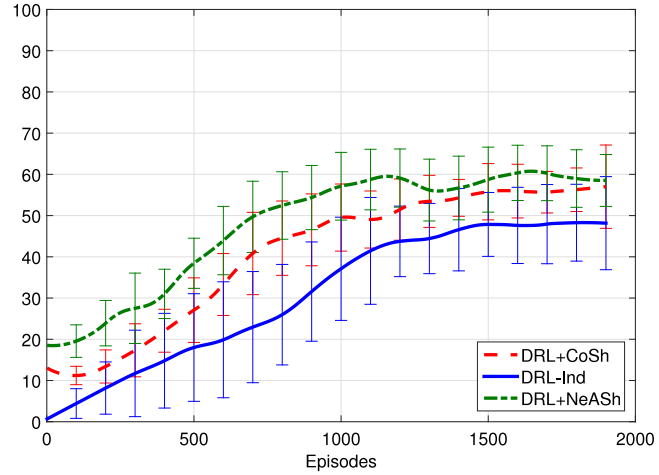

Fig. 6. Inwalk-kicking learning evolution plots for the 3D realistic simulator. Results are averaged across 10 learning runs and error bars show the standard errors.

Table 3
Inwalk-kicking performances (in which 100% is the optimal policy).

Approach	Final performance (%) [Scored Goal Rate]	Time to Th. (40%)
2D Simulator		
DRL+NeASh-SrcHQ	54.55	775
DRL+CoSh-SrcHQ	50.28	915
DRL+NeASh-SrcLQ	49.63	1165
DRL+CoSh-SrcLQ	45.22	1592
DRL-Ind	40.16	1631
3D Simulator		
DRL+NeASh	58.53	531
DRL+CoSh	57.02	688
DRL-Ind	48.16	1064

Table 4
Computational resources comparison between CRL and DRL for the inwalk-kicking problem.

	CRL	DRL
Execution time [ms]	304.72	4.42
Memory usage [MB]	70	0.82

This can be explained taking into account the nearby action effect, because while CoSh always shares the same action for a determined state, NeASh explores the neighborhood according ϱ and $1-\beta$ for sharing a nearby action, which eventually can have a better performance, but surely gives more experience and information to the target DRL agents. As a disadvantage, NeASh requires tuning the extra parameter ϱ.

It is interesting analyzing the effect of those parameters on the knowledge transfer. From Table 1, note that CoSh uses a smaller decay factor to deal with the lower quality in the source ($dec : 19 \rightarrow 13$), which implies a slower learning. NeASh reacts similarly: it reduces dec from 29 to 12, but increases the nearby action deviation by reducing the scale factor ϱ from 25 to 16. This means, if the source is good, that NeASh trusts more in the SoK, but if the source is weak, that NeASh should explore more around the suggested action from the source.

Fig. 6 presents learning evolution plots for the 3D simulator experiments, in which the high quality sources were used. These plots validate results from previous experiments: DRL-NeASh is again the best and fastest scheme (around 10% better and 27% faster than DRL-Ind) followed by DRL+CoSh schemes (around 9% better and 19% faster than DRL-Ind). From Table 1, note that ϱ and dec differ from 2D experiments due to the more challenging and realistic environment. For instance, DRL-NeASh now uses $\varrho = 10$, which is a reduced value compared to the 2D experiment, in order to increase the exploration zone from the source actions.

Finally, it is also interesting to compare numerically the computational benefits proper of the use of DRL instead of CRL. Table 4 presents both the computational speedup and the memory consumption reduction obtained for the case of inwalk-kicking. It is important to note that vanilla CRL produces impractical execution times even for relatively low-dimensional problems such as inwalk-kicking, whereas DRL is able to provide a considerable speedup, allowing its use in real-time applications. A similar phenomena is presented in the memory consumption. While the memory usage of the CRL approach for this problem is still feasible to nowadays computers, it still impossible to deploy such solutions to embedded systems. In this case, DRL also proves to address this issue, producing models which are able to be embedded in such platforms, due to a much reduced memory usage. A more detailed CRL vs. DRL comparison and analysis for this problem has been addressed by Lobos-Tsunekawa et al. (2017).

4.2. Ball-dribbling

Ball-dribbling is a complex behavior during which a robot player attempts to maneuver the ball in a very controlled way, while moving

toward a desired target. Used variables are the same for the inwalk-kicking problem, described at the beginning of this section and shown in Fig. 3.

4.2.1. Decentralized modeling

As in the inwalk-kicking problem, the NAO robot is used. So, the same three separate agents $Agent_x$, $Agent_y$, and $Agent_\theta$ are proposed. Our expected common goal is to walk fast toward the desired target while keeping possession of the ball. That means: to maintain $\rho < \rho_{th}$; to minimize $\gamma, \phi, v_y, v_\theta$; and to maximize v_x. In this way, this ball-dribbling behavior can be separated into three tasks or individual goals, which have to be executed in parallel: *ball-turning*, which keeps the robot tracking the ball-angle ($\gamma = 0$); *alignment*, which keeps the robot aligned to the ball-target line ($\phi = 0$); and *ball-pushing*, whose objective is for the robot to walk as fast as possible and hit the ball in order to change its speed, without losing possession of it. So, the proposed control signals are $[v_x, v_y, v_\theta]$, respectively, involved with *ball-pushing*, *alignment*, and *ball-turning*. Thus, individual rewards are proposed for each learning agent:

$$r^x = \begin{cases} 1, & \text{if } (\rho < \rho_{th}) \wedge (\gamma < \gamma_{th}) \wedge (\phi < \phi_{th}) \wedge (v_x < v_{x.max'}), \\ -1, & \text{otherwise,} \end{cases}$$
$$r^y = \begin{cases} 1, & \text{if } (\gamma < \gamma_{th}/3) \wedge (\phi < \phi_{th}/3), \\ -1, & \text{otherwise,} \end{cases} \quad (6)$$
$$r^\theta = \begin{cases} 1, & \text{if } (\gamma < \gamma_{th}/3) \wedge (\phi < \phi_{th}/3), \\ -1, & \text{otherwise,} \end{cases}$$

where: $[\rho_{th}, \gamma_{th}, \phi_{th}]$ are desired thresholds at which the ball is considered to be controlled, otherwise a *fault-state* occurs; and $v_{x.max'}$ reinforces walking forward at maximum speed. Fault-state thresholds are set as: $[\rho_{th}, \gamma_{th}, \phi_{th}] = [250\text{ mm}, 25°, 25°]$, and $v_{x.max'} = 0.9 \cdot v_{x.max}$. The complete proposed modeling for learning the 3-Dimensional velocity vector from the joint observed state is detailed in Table 2.

4.2.2. Experimental setup

A SARSA(λ) RL algorithm with RBF approximation is also implemented for these experiments. Parameters and decayed functions are set and configured in the same way as for the kicking problem. All the parameter are detailed in Table 1 for each scheme implemented.

The ball-dribbling RL procedure is carried out episodically and 1000 episodes are trained in the SimRobot 3D simulator. After a reset, the robot is set near to its own goal (Fig. 4, right), in a random position over the red arc around the ball, and the desired target is defined by $\oplus$. The terminal state is reached if the robot loses the ball, if the robot leaves the field, or if the robot reaches the target (which is the expected terminal state). The training field is 9 × 6 m.

The evolution of the learning process is evaluated by measuring and averaging 10 runs. In this way, the following performance indices are considered to measure *dribbling-speed* and *ball-control* respectively:

- *% of maximum forward speed* ($\%S_{Fmax}$): given S_{Favg}, the average dribbling forward speed of the robot, and S_{Fmax}, the maximum forward speed $\%S_{Fmax} = S_{Favg}/S_{Fmax}$. $\%S_{Fmax} = 100\%$ is the best performance.
- *% of time in fault-state* ($\%T_{FS}$): is the accumulated time in *fault-state* t_{FS} during the whole episode time t_{DP}. The *fault-state* is defined as the state when the robot loses possession of the ball, i.e., $\rho > \rho_{th} \vee |\gamma| > \gamma_{th} \vee |\phi| > \phi_{th}$, then $\%T_{FS} = t_{FS}/t_{DP}$. $\%T_{FS} = 0$ is the best performance.
- *Global Fitness (F):* computed as $F = 1/2[(100\%S_{Fmax}) + \%T_{FS}]$, where $F = 0$ is the optimal but non-reachable policy.

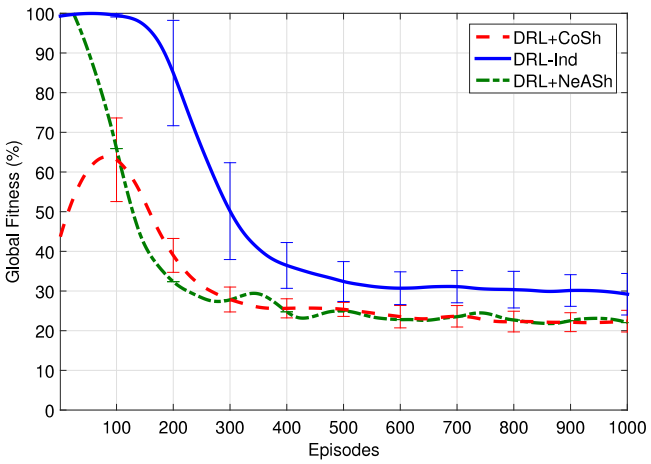

Fig. 7. Ball-dribbling learning evolution plots for the 3D realistic simulator. Results are averaged across 10 learning runs and error bars show the standard errors.

Table 5
Ball-dribbling performances (in which lower %s are better).

Approach	Final performance (%) [Scored Goal Rate]	Time to Th. (30.0%)
DRL+NeASh	22.04	225
DRL-CoSh	22.40	267
DRL-Ind	29.19	845
RL-FLC	34.40	51

4.2.3. Results and analysis

Fig. 7 shows the learning evolution plots and Table 5 shows the averaged final performances and learning times for achieving a performance of 30%. DRL+NeASh scheme shows the best final performance and learning times, around 7% better and 62% faster than DRL-Ind, followed by DRL+CoSh scheme which is around 6% better and 58% faster than DRL-Ind. DRL-Ind shows the lowest performance, slower learning times, and larger error bars with respect to the TK approaches. Same as in the inwalk-kicking problem, it is expected due to the lack of prior-coordination in the DRL-Ind scheme, contrary to the DRL+CoSh and DRL+NeASh schemes, which for this experiments used a SoK with a prior performance of around 45%.

Since a previous implementation for the ball-dribbling problem has been already reported in the literature as RL-FLC (Leottau et al., 2015), we have included its performance indices in Table 5. The effectiveness and benefits of this hybrid RL and fuzzy approach have been pointed out by Leottau et al. (2015). However, a significant human effort and knowledge of the controller designer are required for implementing all the proposed stages. In that sense, our independent DRL approach is able to learn the whole ball-dribbling behavior autonomously, achieving best performances with respect to the RL-FLC with less human effort and less previous knowledge. An advantage that still remains from the RL-FLC method is the considerably lower RL training time, regarding the DRL scheme (51 episodes vs. 845 episodes approximately for achieving a performance of 30%). In that sense, the transfer knowledge strategies for DRL agents proposed in this work are able to reduce that learning time down up to 225 episodes, opening the door to make achievable future implementations for learning similar behaviors with physical robots.

5. Conclusions

This paper presented a Decentralized Reinforcement Learning (DRL) architecture to alleviate the effects of the curse of dimensionality and the large number of training trials required to learn tasks in which multi-dimensional action spaces are involved. Three DRL schemes are

considered and tested: DRL-Ind, implemented with independent learners and no prior-coordination; DRL+CoSh, accelerated-coordinated by using the Control Sharing (CoSh) knowledge transfer approach, which is extended from the single-agent case to the DRL proposed architecture; and DRL+NeASh, a knowledge transfer approach proposed for including a measure of uncertainty to the CoSh procedure.

The proposed methods have been validated by implementing two real-world problems, the inwalk-kicking and the ball-dribbling behaviors, both performed with humanoid biped robots, where each component of the requested velocity vector $[v_x, v_y, v_\theta]$ is learned in parallel with independent agents working in a multi-agent task. Results have shown empirically that benefits of MAS are also applicable to complex problems like robotic platforms, by using a DRL architecture. Results have also shown that even without prior-coordination, both asymptotic convergence and indirect coordination are achieved among DRL-Ind agents. They have shown that it is possible to reduce the training episodes and coordinate the DRL by using knowledge transfer from simple linear controllers, obtaining better performances and learning times with respect to the DRL-Ind scheme.

DRL+NeASh schemes showed either better performances and learning times for the inwalk-kicking problem. By contrast, DRL+NeASh and DRL+CoSh approaches showed similar performances for the dribbling problem. This is an interesting point to be discussed taking into account the quality of the sources of knowledge used by each problem: the inwalk-kicking behavior used a source of knowledge with a performance of around 15% (being 100% the optimal), while the ball-dribbling used a source of knowledge of around 45% (being 0% the optimal). The DRL+NeASh scheme showed the best averaged performance for both problems: 58.53% for the inwalk-kicking, and 22.04% for the ball-dribbling. Note that the source performance for the dribbling case was closer to the optimal policy. Thus, we can empirically conclude that benefits of NeASh approach are more noticeable when the source of knowledge has poor performances or more uncertainty; otherwise, the CoSh approach could be more convenient due to its simplicity and easy parameter tuning, and also because of CoSh is able to deal with both, discrete and continuous action spaces.

Video demonstrating the inwalk-kick and ball-dribbling learned policies performed with a real NAO robot can be found online at Lobos-Tsunekawa (2017). The policies are transferred directly to the physical robot, thus, the final performance is dependent on how realistic the simulation platform is.

CoSh and NeASh were able to fulfill all the requirements indicated in Section 3.2: transferring on any RL algorithm that uses an action value function; transferring from different SoK types; including prior-coordination without more complexity than a single-agent RL method; and allowing heterogeneous agents with different source and target tasks but same source and target problem-space. Those considerations were accomplished precisely because of the NeASh and CoSh simplicity. However, since this work is one of the first approaches that address coordination or acceleration of DRL systems, our intention was to introduce basic and simple concepts and methods as a starting point, and as motivation for future researches on this field, in which more sophisticated methods can be used.

Acknowledgments

This research was partially funded by FONDECYT Project, Chile 1161500. David Leonardo Leottau was funded under grant CONICYT-PCHA/Doctorado Nacional, Chile/2013-63130183. Francisco Jaramillo was funded under grant CONICYT-PCHA/Doctorado Nacional, Chile/2014-21140201.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.06.019.

References

Bianchi, R.A.C., Martins, M.F., Ribeiro, C.H.C., Costa, A.H.R., 2014. Heuristically-accelerated multiagent reinforcement learning. IEEE Trans. Cybern. 44, 252–265.

Boutsioukis, G., Partalas, I., Vlahavas, I., 2012. Transfer learning in multi-agent reinforcement learning domains. In: Sanner, S., Hutter, M. (Eds.), Recent Advances in Reinforcement Learning. Springer Berlin Heidelberg, pp. 249–260.

Buşoniu, L., Babuška, R., De Schutter, B., 2008. A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. C 38, 156–172.

Buşoniu, L., De Schutter, B., Babuška, R., 2006. Decentralized reinforcement learning control of a robotic manipulator. In: Ninth International Conference on Control, Automation, Robotics and Vision, ICARCV. Singapore, pp. 1–6.

Claus, C., Boutilier, C., 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. In: AAAI '98/IAAI '98, Madison, Wisconsin, USA, pp. 746–752.

Dziomin, U., Kabysh, A., Golovko, V., Stetter, R., 2013. A multi-agent reinforcement learning approach for the efficient control of mobile robot. In: 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). Berlin, Germany, pp. 867–873.

Fernández, F., García, J., Veloso, M., 2010. Probabilistic policy reuse for inter-task transfer learning. Robot. Auton. Syst. 58, 866–871.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B., 2009. Mechatronic design of NAO humanoid. In: 2009 IEEE International Conference on Robotics and Automation. Kobe, Japan, pp. 769–774.

Hu, Y., Gao, Y., An, B., 2015. Accelerating multiagent reinforcement learning by equilibrium transfer. IEEE Trans. Cybern. 45, 1289–1302.

Kabysh, A., Golovko, V., Lipnickas, A., 2012. Influence learning for multi-agent system based on reinforcement learning. Int. J. Comput. 11, 39–44.

Knox, W.B., Stone, P., 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1, AAMAS 2010. International Foundation for Autonomous Agents and Multiagent Systems, Toronto, Canada, pp. 5–12.

Knox, W.B., Stone, P., 2012. Reinforcement learning from simultaneous human and MDP reward. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS 2012. International Foundation for Autonomous Agents and Multiagent Systems, pp. 475–482.

Konidaris, G., Scheidwasser, I., Barto, A.G., 2012. Transfer in reinforcement learning via shared features. J. Mach. Learn. Res. 13, 1333–1371.

Laurent, G.J., Matignon, L., Le Fort-Piat, N., 2011. The world of independent learners is not Markovian. Int. J. Knowl.-Based Intell. Eng. Syst. 15, 55–64.

Leottau, D.L., Celemin, C., Ruiz-del Solar, J., 2015. Ball dribbling for humanoid biped robots: A reinforcement learning and fuzzy control approach. In: Bianchi, R.A.C., Akin, H.L., Ramamoorthy, S., Sugiura, K. (Eds.), RoboCup 2014: Robot World Cup XVIII. In: Lecture Notes in Computer Science, vol. 8992, Springer Verlag, Berlin, pp. 549–561.

Leottau, D.L., Ruiz-Del-Solar, J., 2015. An accelerated approach to decentralized reinforcement learning of the ball-dribbling behavior. In: AAAI Workshops. Austin, Texas USA, pp. 23–29.

Leottau, D.L., Ruiz-del Solar, J., Babuška, R., 2018. Decentralized reinforcement learning of robot behaviors. Artificial Intelligence 256, 130–159.

Leottau, D.L., Ruiz-del Solar, J., MacAlpine, P., Stone, P., 2016. 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. In: Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin, pp. 290–302.

Leottau, D.L., Vatsyayan, A., Ruiz-del Solar, J., Babuška, R., 2017. Decentralized reinforcement learning applied to mobile robots. In: Behnke, S., Sheh, R., Sariel, S., Lee, D. (Eds.), RoboCup 2016: Robot World Cup XX. In: Lecture Notes in Artificial Intelligence, vol. 9776, Springer Verlag, Berlin.

Lobos-Tsunekawa, K., 2017. Inwalk-kicking and ball-dribbling videos. https://drive.google.com/drive/folders/1t68DUVdXKbf-C65EmBHEdsVGrqGbo7W2?usp=sharing (accessed 7.2.18).

Lobos-Tsunekawa, K., Leottau, D.L., Ruiz-del Solar, J., 2017. Toward real-time decentralized reinforcement learning using finite support basis functions. CoRR abs/1706.06695.

Martín, J.A., de Lope Asiaín, J., 2007. A distributed reinforcement learning control architecture for multi-link robots - experimental validation. In: Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics, ICINCO 2007. Angers, Francia, pp. 192–197.

Mataric, M.J., 1994. Reward functions for accelerated learning. In: Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann, Boca Raton, Florida, USA, pp. 181–189.

Papierok, S., Noglik, A., Pauli, J., 2008. Application of reinforcement learning in a real environment using an RBF network. In: 1st International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems (ERLARS 2008). Patras, Greece, pp. 17–22.

Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F., 2011. B-Human Team Report and Code Release 2011. Department of Computer Science, University of Bremen, Bremen, Germany.

Schuitema, E., 2012. Reinforcement Learning on Autonomous Humanoid Robots (Ph.D. Thesis). Delft University of Technology, p. 195.

Sutton, R., Barto, A., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

Takahashi, Y., Asada, M., 2005. Multi-layered learning system for real robot behavior acquisition. In: Kordic, V., Lazinica, A., Merdan, M. (Eds.), Cutting Edge Robotics. Intech, Germany, pp. 357–375.

Taylor, A., Dusparic, I., Cahill, V., 2013. Transfer learning in multi-agent systems through parallel transfer. In: 30th International Conference on Machine Learning. Atlanta, USA.

Taylor, M., Stone, P., 2007. Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th International Conference on Machine Learning - ICML 2007. ACM Press, New York, New York, USA, pp. 879–886.

Taylor, M., Stone, P., 2009. Transfer learning for reinforcement learning domains: A survey. J. Mach. Learn. Res. 10, 1633–1685.

Troost, S., Schuitema, E., Jonker, P., 2008. Using cooperative multi-agent Q-learning to achieve action space decomposition within single robots. In: 1st International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems (ERLARS 2008). Patras, Greece, pp. 23–32.

Veloso, M., Stone, P., 2012. Video: RoboCup robot soccer history 1997-2011. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal, pp. 5452–5453.

Vrancx, P., De Hauwere, Y., Nowé, A., 2011. Transfer learning for multi-agent coordination. In: The 3th International Conference on Agents and Artificial Intelligence. Rome, Italy, pp. 263–272.

Yáñez, J.M., Cano, P., Mattamala, M., Leottau, D.L., Celemín, C., Saavedra, P., Villegas, C., Lobos, K., Azócar, G., Cruz, N., Pérez, R., Miranda, P., Verdugo, C., Herrera, F., Cossio, L., Ruiz-del-Solar, J., 2016. UChile robotics team team description for RoboCup 2016. In: RoboCup 2016: Robot Soccer World Cup XX Preproceedings, July 2016. Leipzig, Germany.