Reinforcement Learning with Case-Based Heuristics for RoboCup Soccer Keepaway

Luiz A. Celiberto Jr. and Jackson P. Matsuura
Technological Institute of Aeronautics
São José dos Campos, Brasil.
celibertojr@gmail.com, jackson@ita.br

Ramón López de Màntaras
Artificial Intelligence Research Institute
Spanish National Research Council
Bellaterra, Spain.
mantaras@iiia.csic.es

Reinaldo A. C. Bianchi
Centro Universitário da FEI
São Bernardo do Campo, Brasil.
rbianchi@fei.edu.br

***Abstract*—In this paper we propose to combine Case-based Reasoning and Heuristically Accelerated Reinforcement Learning to speed up a Reinforcement Learning algorithm in a Transfer Learning problem. To do so, we propose a new algorithm called SARSA Accelerated by Transfer Learning – SATL, which uses Reinforcement Learning to learn how to perform one task, stores the policy for this problem as a case-base and then uses the learned case-base as heuristics to speed up the learning performance in a related, but different, task. A set of empirical evaluations were conducted in transferring the learning between two domains with multiple agents: an expanded version of Littman's simulated robot soccer and the RoboCup Soccer Keepaway. A policy learned by one agent in the Littman's soccer is used to speed up the agent learning in the Keepaway soccer. The results show that the use of this new algorithm can lead to a significant improvement in the performance of the learning agents.**

***Keywords*-Artificial Intelligence; Machine Learning; Transfer Learning; Knowledge based systems**

I. Introduction

Reinforcement Learning (RL) [22] is a paradigm that is very successful when used by agents trying to maximize some notion of cumulative reward, as they carry out sensing, decision, and action in an unknown environment. Unfortunately, RL tasks encountered in the real world commonly suffer from very slow learning rates, with the basic learning algorithms have often requiring a large number of iterations to converge on a good solution.

Much of the prevailing investigation in the field focuses on how to speed up RL by taking advantage of domain knowledge. One way to do this is by making use of a heuristic function that encodes some expertise, which is used for selecting appropriate actions to perform in order to guide exploration during the learning process [2]. Several methods have been successfully applied for defining the heuristic function, including the reuse of previously learned policies, using a Case-Based Reasoning approach [3]. Another way to speed up a RL using domain knowledge is by using Transfer Learning techniques: "Transfer learning can increase RL's applicability to difficult tasks by allowing agents to generalize their experience across learning problems" [25].

This paper investigates the use of a case-base as a heuristic to transfer the learning acquired by one agent during its training in one problem to another agent that has to learn how to solve a similar, but more complex, problem. To do so we propose a new algorithm, the SARSA Accelerated by Transfer Learning – SATL, which is a Case-based, Heuristically Accelerated extension of the traditional RL algorithm, SARSA [17].

Experiments in this work were conducted in two domains: the source domain, a version of the Littman's simulator for robot soccer [11] modified to allow games with two or more agents in each team; and the target domain, the RoboCup Soccer Keepaway problem [21].

The paper is organized as follows: Section II briefly reviews the RL problem and describes the HARL approach to speed up RL. Section III describes the Case Based Reasoning technique and Section IV describes the Transfer Learning problem. Section V describes the proposed algorithm and section VI describes the experiment and result. Section VII concludes this work.

II. Heuristically Accelerated Reinforcement Learning

Reinforcement Learning (RL) algorithms have been applied successfully to the on-line learning of optimal control policies in Markov Decision Processes (MDPs). In RL, this policy is learned through trial-and-error interactions of the agent with its environment: on each interaction step the agent senses the current state s of the environment, chooses an action a to perform, executes this action, altering the state s of the environment, and receives a scalar reinforcement signal r (a reward or penalty).

The RL problem can be formulated as a discrete time, finite state, finite action Markov Decision Process (MDP). The learning environment can be modeled by a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where:

- $\mathcal{S}$: is a finite set of states.
- $\mathcal{A}$: is a finite set of actions that the agent can perform.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$: is a state transition function, where $\Pi(\mathcal{S})$ is a probability distribution over $\mathcal{S}$. $T(s, a, s')$ represents the probability of moving from state s to s' by performing action a.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$: is a scalar reward function.

The goal of the agent in a RL problem is to learn an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that maps the current state s into the most desirable action a to be performed in s. One strategy to learn the optimal policy π^* is to allow the agent to learn the evaluation function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. Each

Table I
THE SARSA ALGORITHM [17].

Initialize $\hat{Q}_t(s,a)$ arbitrarily.
Repeat (for each episode):
  Initialize s.
  Repeat (for each step):
    Select an action a.
    Execute the action a, observe $r(s,a)$, s'.
    Update the values of $Q(s,a)$ using 2.
    $s \leftarrow s'$.
  Until s is terminal.
Until some stopping criterion is reached.

Table II
THE HA-SARSA ALGORITHM.

Initialize $\hat{Q}_t(s,a)$ and $H_t(s,a)$ arbitrarily.
Repeat (for each episode):
  Initialize s.
  Repeat (for each step):
    Update the values of $H_t(s,a)$ as desired.
    Select an action a using equation 3.
    Execute the action a, observe $r(s,a)$, s'.
    Update the values of $Q(s,a)$ according to equation 2.
    $s \leftarrow s'$.
  Until s is terminal.
Until some stopping criterion is reached.

action value $Q(s,a)$ represents the expected cost incurred by the agent when taking action a at state s and following an optimal policy thereafter. Several algorithms have been proposed to learn the optimal policy, like Q–Learning [30], SARSA and SARSA(λ) [17], Dyna [23] and Prioritized Sweeping [13], [15]. A compreensive review of RL algorithms can be found in Sutton and Barto [22].

The Q–learning algorithm was proposed by Watkins [30] as a strategy to learn an optimal policy π^* when the model ($\mathcal{T}$ and $\mathcal{R}$) is not known in advance. Let $\hat{Q}$ be the learner's estimate of $Q^*(s,a)$. The Q–learning algorithm iteratively approximates $\hat{Q}$ provided the system can be modeled as a MDP, the reward function is bounded ($\exists c \in \mathcal{R}; (\forall s,a), |R(s,a)| < c$), and actions are chosen so that every state-action pair is visited an infinite number of times. The Q learning update rule is:

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

where s is the current state; a is the action performed in s; r is the reward received; s' is the new state; γ is the discount factor ($0 \leq \gamma < 1$); $\alpha = 1/(1 + visits(s,a))$, where $visits(s,a)$ is the total number of times this state-action pair has been visited up to and including the current iteration.

The SARSA algorithm was proposed by Rummery and Niranjan [17] as another way to learn an optimal policy π^*. The main difference between the two algorithms is that the SARSA is a on-policy control algorithm, while the Q–Learning is an off-line method. The algorithm is presented inTable I, using the following update rule is:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \left[r + \gamma \hat{Q}(s',a') - \hat{Q}(s,a) \right], \quad (2)$$

In RL, learning is carried out online, through trial-and-error interactions of the agent with the environment. Unfortunately, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space. In the next section we show one way to speed up the convergence of RL algorithms, by making use of a heuristic function in a manner similar to the use of heuristics in informed search algorithms.

A Heuristically Accelerated Reinforcement Learning (HARL) algorithm [2] is a way to solve a MDP problem with explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ for influencing the choice of actions by the learning agent. $H(s,a)$ defines the heuristic that indicates the importance of performing action a when visiting state s. The heuristic function is strongly associated with the policy indicating which action must be taken regardless of the action-value of the other actions that could be used in the state.

The first HARL algorithm proposed was the Heuristically Accelerated Q–learning (HAQL) [2], as an extension of the Q–learning algorithm [30]. The only difference between the two algorithms is that in the HAQL makes use of a heuristic function $H(s,a)$ in the $\epsilon-greedy$ action choice rule, that can be written as:

$$\pi(s) = \begin{cases} \arg\max_a \left[\hat{Q}(s,a) + \xi H(s,a)^\beta \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \quad (3)$$

where $H(s,a)$ is the heuristic function that plays a role in the action choice, ξ and β are design parameters that control the influence of the heuristic function, q and p are parameters that define the exploration/exploitation tradeoff and a_{random} is an action randomly chosen among those available in state s.

As a general rule, the value of $H(s,a)$ used in HAQL should be higher than the variation among the $\hat{Q}(s,a)$ values for the same $s \in \mathcal{S}$, in such a way that it can influence the choice of actions, and it should be as low as possible in order to minimize the error. It can be defined as:

$$H(s,a) = \begin{cases} \max_i \hat{Q}(s,i) - \hat{Q}(s,a) + \eta \text{ if } a = \pi^H(s), \\ 0 \text{ otherwise.} \end{cases} \quad (4)$$

where η is a small real value (usually 1) and $\pi^H(s)$ is the action suggested by the heuristic policy.

Convergence of the HAQL algorithm was presented by Bianchi, Ribeiro and Costa [2], together with the definition of an upper bound for the error in the estimation of Q.

Using heuristics to speed up Reinforcement Learning algorithms has been shown to be effective. Several works have used heuristics to speed up learning in problems such as Transfer Learning [4], [5], task allocation [10] and especially in the domain of the robotic soccer competitions [1], [6], [7].

Despite these works, the Heuristically Accelerated SARSA is an algorithm that has not been proposed or implemented yet. It's implementation is very similar to

the HAQL, with one minor difference: the use of Equation 2 instead of Equation 1. The complete HA-SARSA algorithm is presented in Table II.

III. Case Based Reasoning

According to López de Mántaras *et al.* [12], solving a problem by CBR involves "obtaining a problem description, measuring the similarity of the current problem to previous problems stored in a case base with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution of the retrieved case(s), possibly after adapting it to account for differences in problem descriptions".

Case Based Reasoning [12] is an AI technique that has been shown to be useful in a multitude of domains. CBR uses knowledge of previous situations (cases) to solve new problems, by finding a similar past case and reusing it in the new problem situation. In the CBR approach, a case usually describes a problem and its solution, i.e., the state of the world in a given instant and action to perform to solve that problem.

In general, in CBR a case is composed of a problem description (P) and the corresponding description of the solution (A). Therefore, the case definition is formally described as a tuple:

$$case = (P, A). \tag{5}$$

The case retrieval process consists in obtaining from the base the most similar case, the retrieved case. Therefore, it is necessary to compute the similarity between the current problem and the cases in the base. The similarity function indicates how similar a problem and a case are. In this work this function is defined by a Gaussian distance between the problem and the case

IV. Transfer Learning

Although the idea of using Transfer Learning to reuse the learning of one task in another related domain is not a new (it has been studied in the psychological literature on transfer of learning since the work of Thorndike and Woodworth [27]), only recently the use of Transfer Learning for Reinforcement Learning has gained attention in the artificial intelligence community.

Is possible to divided the TL in two main categories: intra-domain transfer, where TL is used to solve a new tasks within a given domain and cross-domain transfer where transfer is made between domains. Usually, Intra-domain transfer uses the same space state and transforms primitive actions in more complex actions to be uses in new tasks within this domain. Cross-domain transfer try to find similar structure between the source and target task to transfer the learning.

However different assumptions methods in transfer learning domains may have, an RL agent must at least perform the following steps [25]: Given a target task, select an appropriate source task or set of tasks from which to transfer; Learn how the source task(s) and target task are related; Effectively transfer knowledge from the source task(s) to the target task.

Table III
THE SATL ALGORITHM

Initialize $\hat{Q}_t(s, a)$ and $H_t(s, a)$ arbitrarily.
Repeat (for each episode):
  Initialize s.
  Repeat (for each step):
    Compute similarity and cost.
    If there is a case that can be reused:
      Retrieve and Adapt if necessary.
      Compute $H_t(s, a)$ using Equation 4 with the actions suggested by the case selected.
    Select an action a using equation 3.
    Execute the action a, observe $r(s, a)$, s'.
    Update the values of $Q(s, a)$ using 2.
    $s \leftarrow s'$.
  Until s is terminal.
Until some stopping criterion is reached.

Transfer Learning is a very important tool to speed up RL algorithms because, in RL, even a small change on the configuration of a problem may requires a complete new training. With TL, what an agent has learned can be transferred to a new situation, helping it to learn faster. Drummond [8] was probably the first to use CBR to speed up RL, proposing to accelerate RL by transferring parts of previously learned solutions to a new problem, exploiting the results of prior learning to speed up the process.

V. SARSA Accelerated by Transfer Learning

To transfer the cases between learning agents in two domains, in this work we propose a new algorithm that expands the SARSA by making use of Transfer Learning, Case-based reasoning and heuristics, the SARSA Accelerated by Transfer Learning – SATL – algorithm.

This algorithm works in 2 phases: first, it is used to learns how to perform a task in the source domain, behaving like the traditional SARSA algorithm but also storing the policy for solving this problem as a case-base; in the second phase it uses the case-base learned in the first stage as heuristics in an heuristically accelerated version of the SARSA algorithm, the SATL.

The main motivation of using cases as heuristics to transfer the learning is that the heuristic function is an action policy modifier which does not interfere with the standard bootstrap like update mechanism of RL the algorithm: the SATL differs from the SARSA only in the way exploration is carried out, which allows many theoretical conclusions obtained for the SARSA to remain valid for the SATL.

Similar to the model proposed by Ros [16], the cases used in this work are described by a 3-tuple: $case = (P, A, R)$ where: P is the description of the problem, containing all relevant information of the agent state (a state $s \in \mathcal{S}$); A is an action (or a sequence of actions) that must be performed to solve the problem and; R is the expected return for performing the action, which indicates the quality of the action stored in this case.

Case retrieval is in general driven by a similarity measure between the new problem and the solved problems in the case base. In this work we use the case retrieval

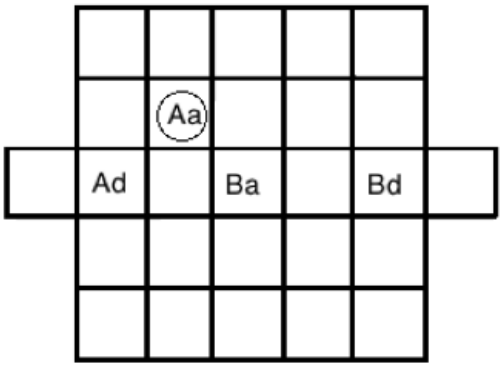

Figure 1. The modified Littman's Soccer domain used in this work.

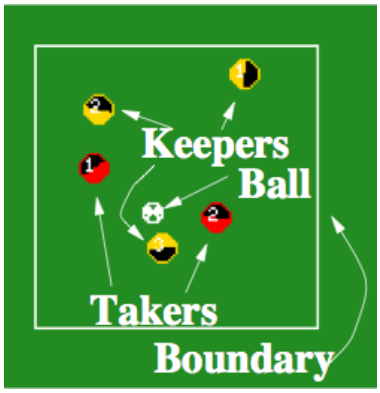

Figure 2. The Keepaway Soccer domain, with 3 keepers and 2 takers [20].

method proposed by Ros [16], which considers the similarity between the problem and the case (the similarity is computed using by Gaussian distance between the case and the problem), the cost of adapting the problem to the case, and the applicability of the solution of the case. The cost of adapting the problem to the case is computed as a function of the euclidean distances between the features (positions of players and ball) in the problem and the ones specified in the case. The complete case retrieval algorithm is described in detail in Ros [16].

After a case is retrieved, a heuristic is computed using Equation 4 and the action suggested by the case is selected and executed. If the case base does not contain a case that can be used in the current situation, the SATL algorithm will behave as the traditional SARSA algorithm.

Our approach differs from previous research combining CBR and RL because the policy learned in one domain is stored as a case base and then used in a new domain as a heuristic: it is used in the action selection rule to guide the search in the new domain, in the same way a heuristic is used in an informed search method. At the beginning of each learning episode, RL operates as a blind search method does: in our view, cases can be used to improve RL from a blind search method to an informed search one. By doing this, if the case base contains a case that can be used in a given situation, then there will be a speed up in the convergence time. But if the case base does not contain any useful case – or even if it contains cases that implement wrong solutions to the problem – the agent will still learn the optimal solution by using the RL component of the algorithm. The complete SATL algorithm is presented in Table III.

VI. One Experiment using the SATL

In the this section we present an experiment using the SATL algorithm, where cases acquired in the source domain, the Littman's simulator for robot soccer [11] modified to allow games with two or more agents in each team (as proposed in [1]), are used to speed up the learning of the RoboCup Soccer Keepaway domain [21].

The modified Littman's Soccer domain used in this work is a game played by two teams, A and B, of two players each, wich compete in a 5 x 5 grid (figure 1). Each team is composed by the defender (d) and the attacker (a). Each cell can be occupied by one of the players, which can take an action at a turn. The actions that are allowed are: keep the agent still, move – north, south, east and west – or pass the ball to another agent. The action "pass the ball" from agent a_i to a_j is successful if there is no opponent in between them. If there is an opponent, it will catch the ball and the action will fail.

Actions are taken in turns: all actions from one team's agents are executed at the same instant, and then the opponents actions are executed. The ball is always with one of the players. When a player executes an action that would finish in a cell occupied by the opponent, it looses the ball and stays in the same cell. If an action taken by the agent leads it out the board, the agent stands still. When a player with the ball gets into the opponent's goal, the move ends and its team scores one point. At the beginning of each game, the agents are positioned in a random position and the possession of the ball is randomly determined, with the player that holds the ball making the first move.

The target domain, the Keepaway Soccer, is a subproblem of RoboCup simulated soccer in which one team, the keepers, tries to maintain possession of the ball within a limited region, while the opposing team, the takers, attempts to gain possession. "Whenever the takers take possession or the ball leaves the region, the episode ends and the players are reset for another episode (with the keepers being given possession of the ball again). Parameters of the task include the size of the region, the number of keepers, and the number of takers" [20].

Figure 2 presents the Keepaway Soccer domain. It is a snapshot of an episode with 3 keepers and 2 takers playing in a 20m x 20m region. Anyone of the Keepers can keep the ball or may pass it to one of its teammates by choosing one of the following macro-actions: Holdball or PassKThenReceive, where K is the first or second closest teammate.

The state variables used for learning with three keepers and two takers is 13 dimension vector, shown in Figure 3. The reward received by the agents is the time that the ball is in possession of the keepers. The takers do not have any kind of learning, and they only choose macro-actions

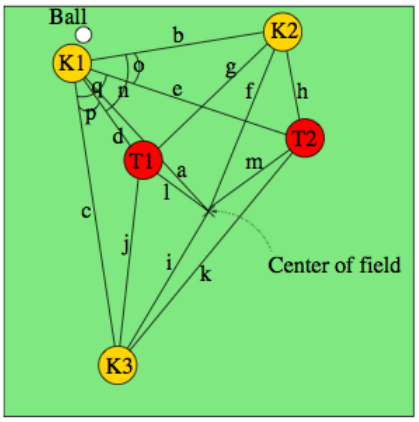

Figure 3. The 13 state variables used by a agent playing with three keepers and two takers [26].

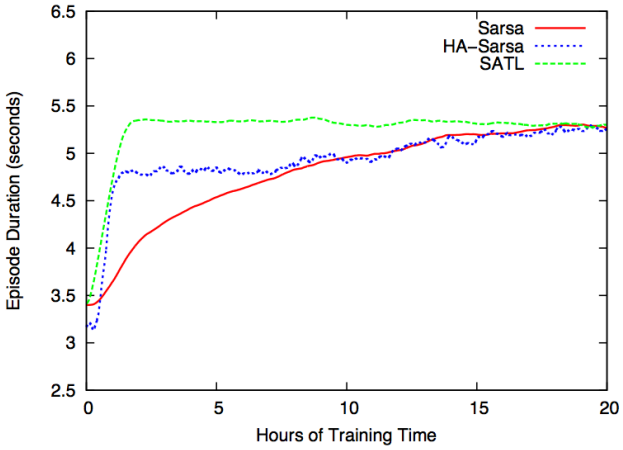

Figure 4. The learning curves for the SARSA, HA-SARSA and SATL algorithms

to block the pass between the keepers, to go to ball or to hold the ball when have its possession.

As explained in the last section, the SATL algorithm works in two phases: learning a task in the source domain, storing the policy as a case-base and using the case-base as heuristics to speed up the learning in the target domain.

Therefore, in the first phase of the experiment, the SARSA algorithm is used in the Littman's domain, with agents continuously playing soccer games. Acquiring cases starts at the beginning of the training. A case is stored every time a player from the opponent team steals the ball, i.e., an agent was holding ball and lost it or tried to pass it to another teammate, losing it. At the end of this training, 600 cases are stored in the case base, and every case represents a situation in which a player lost the ball.

Each case is described by: The problem description (P), which is composed by the distance of the agent to all other players (items 'a' to 'm' in Figure 3); the action (A), that describes the action the player was doing when it lost the ball (holding it or passing the ball) and the expected return (R), that indicates how bad was the result of the action.

In the second phase of the learning, cases acquired in the Littman's soccer domain are transferred and used to speed up the learning in the Keepaway domain. A case is retrieved during the learning if the similarity is above a certain threshold. After a case is retrieved, a heuristic is computed with help of a pre-defined action-mapping (which is trivial in this case, as there are only two actions in both domains, 'Holdball' and 'Passball'). This is achieved by using equation 4 with $\eta = -1$. The heuristic is computed using this negative value, to indicate that the action that lead the agent to loose the ball must not be taken.

Two other algorithms were used in the experiment aimed to verify the hypothesis that the SATL algorithm improves the learning rate: The SARSA algorithm proposed by Rummery and Niranjan [17], which was presented in Section 2 and the Heuristically Accelerated SARSA.

Although the Heuristically Accelerated SARSA has been described as future works in several papers, it is an algorithm that have never been implemented. We use this algorithm in this work to be able to verify if the speed up achieved by the SATL algorithm is caused only by the use of a simple heuristics, or if the use of the case base is an important factor in the speeding up the learning. The heuristic used in the HA-SARSA algorithm was defined using a simple rule: if the agent is holding the ball, pass it to another agent. This is achieved by using $\eta = -1$ and $\pi^H(s) =$ 'HoldBall' in equation 4.

Fifteen training sessions were executed, with each session consisting of thirty hours of learning. Figure 4 show the learning curves for all algorithms. It can be see that the performance of the SARSA is worst than that of the SATL at the initial learning phase; later the performance of the two algorithms become more similar, as expected. It can also be seen that the SATL performs better than the HA-SARSA, indicating that the speed up was not due only to the use of a simple heuristic. The parameters used in the experiments were the same for all the algorithms: exploration/ exploitation$= 0.2$, $\gamma = 0.9$ and $\eta = -1$, to the Littman used $\alpha = 0.9$ and to the Keepaway $\alpha = 0.125$. The reward used in the source somain was $+100$ a goal is made and -100 when the teal concedes a goal. The reward in the target domain is -100 when the ball is lost.

Student's t-Test [19] was used to verify the hypothesis that the transfer of learning speeds up the learning process. According to Nehmzow [14], if two different control programs produced two different means of a particular result, the t-Test can be used to decide whether there is a significant difference between these two means, in order to determine whether one of the two programs produces better results than the other. For the experiments the value of the module of T was computed for each episode using the same data presented in Figure 4. The greater the value of T, more significantly different are the results. The dotted line indicates 99.995% of level of confidence, i.e. results above the line are different and the probability of this statement to be erroneous is less than 1%. The result, presented in Figure 5 shows that SATL performs clearly

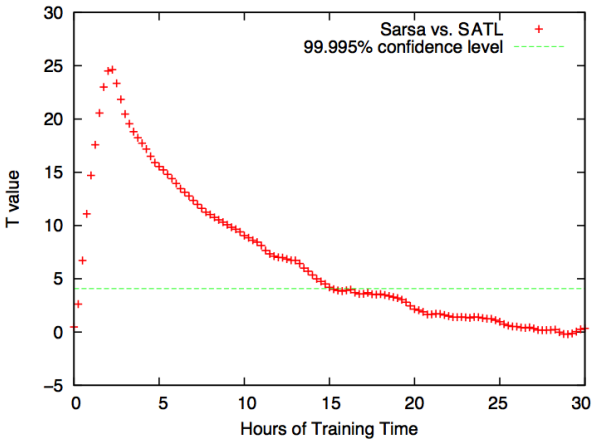

Figure 5. Student's t-Test between SARSA and SATL

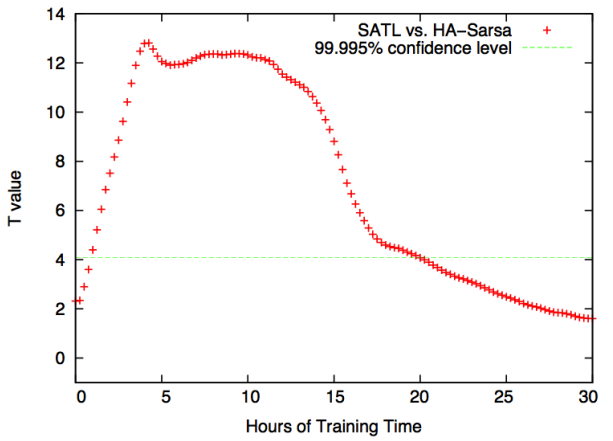

Figure 6. Student's t-Test between HA-SARSA and SATL

better than SARSA until more than 15 hours of training. After that, the results became closer. Similarly, SATL performs better than HA-SARSA, as shown in Figure 6.

VII. CONCLUSION

This work proposed two new algorithms, SATL and HA-SARSA. The first algorithm combines Case-based Reasoning and Heuristically Accelerated Reinforcement Learning to speed up the SARSA algorithm, using Transfer Learning. The second algorithm is the Heuristically Accelerated version of the SARSA algorithm.

This is the first work that implements a Heuristically Accelerated Reinforcement Learning algorithm based on an algorithm that is not the Q–Learning: the SARSA algorithm and studies the transfer of learning using heuristics between domains with multiple agents and this is the first work in which the case-based used to transfer the learning is composed by a set of bad cases, i.e., cases that indicate moments when a wrong decision was made.

The experiments showed that transferring the policy learned by the agents in one domain to agents in a different domain by means of the case-base speeds up the convergence time of the algorithm.

VIII. ACKNOWLEDGMENTS

Reinaldo Bianchi acknowledges the support of the FAPESP (grants 2011/19280-8 and 2012/04089-3). Ramon Lopez de Mantaras acknowledges the grant from Generalitat de Catalunya (2009-SGR-1434) and Luiz Celiberto Jr. acknowledges the support of CAPES.

REFERENCES

[1] Reinaldo A. C. Bianchi and Ramón López de Màntaras, 'Case-Based Multiagent Reinforcement Learning: Cases as heuristics for selection of actions', in *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pp. 355–360, Amsterdam, The Netherlands, The Netherlands, (2010). IOS Press.

[2] Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro, and Anna H. R. Costa, 'Accelerating autonomous learning by using heuristic selection of actions', *Journal of Heuristics*, **14**(2), 135–168, (2008).

[3] Reinaldo A. C. Bianchi, Raquel Ros, and Ramon López de Mántaras, 'Improving reinforcement learning by using case based heuristics', in *Case-Based Reasoning Research and Development, 8th International Conference on Case-Based Reasoning, ICCBR 2009, Seattle, WA, USA, July 20-23, 2009, Proceedings*, eds., Lorraine McGinty and David C. Wilson, volume 5650 of *Lecture Notes in Computer Science*, pp. 75–89. Springer, (2009).

[4] L. A. Celiberto Jr , J. P. Matsuura, R. Lopez de Mantaras and R. A. C. Bianchi, "Using Cases as Heuristics in Reinforcement Learning: A Transfer Learning Application" in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. AAAI Press / IJCAI, 2011. v. 2. p. 1211-1217.

[5] L. A. Celiberto Jr, J. P. Matsuura, R. Lopez de Mantaras and R. A. C. Bianchi, "Using Transfer Learning to Speed-Up Reinforcement Learning: a Cased-Based Approach" in *Proceedings of the 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*, Los Alamitos: IEEE Computer Society, 2010. p. 55-60.

[6] L. A. Celiberto Jr, C. H. C. Ribeiro, A. H. R. Costa and R. A. C. Bianchi "Heuristic Reinforcement Learning Applied to RoboCup Simulation Agents" in *Lecture Notes in Artificial Intelligence*, Berlin : Springer, 2008. v. 5001. p. 220-227.

[7] L. A. Celiberto Jr , J. P. Matsuura and R. A. C. Bianchi "Heuristic Q-Learning Soccer Players: A New Reinforcement Learning Approach to RoboCup Simulation" in *Lecture Notes in Artificial Intelligence*, Berlin : Springer, 2007. v. 4874. p. 520-529.

[8] Chris Drummond, 'Accelerating Reinforcement Learning by composing solutions of automatically identified subtasks', *Journal of Artificial Intelligence Research*, **16**, 59–104, (2002).

[9] Fernando Fernández and Manuela Veloso, 'Probabilistic policy reuse in a Reinforcement Learning agent', in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pp. 720–727, New York, NY, USA, (2006). ACM.

[10] J. A. Gurzoni Jr, F. Tonidandel and R. A. C. Bianchi "Market-Based Dynamic Task Allocation using Heuristically Accelerated Reinforcement Learning" in *Lecture Notes in Artificial Intelligence*, Berlin : Springer, 2011. v. 7026. p. 365-376.

[11] Michael L. Littman, 'Markov games as a framework for multi-agent reinforcement learning', in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163. Morgan Kaufmann, (1994).

[12] Ramon López de Mántaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael T. Cox, Kenneth Forbus, Mark Keane, Agnar Aamodt, and Ian Watson, 'Retrieval, reuse, revision and retention in case-based reasoning', *Knowl. Eng. Rev.*, **20**(3), 215–240, (2005).

[13] Andrew W. Moore and Christopher G. Atkeson, 'Prioritized sweeping: Reinforcement learning with less data and less time', *Machine Learning*, **13**, 103–130, (1993).

[14] Ulrich Nehmzow, *Scientific Methods in Mobile Robotics: quantitative analysis of agent behaviour.*, Springer-Verlag London Limited, London, 2006.

[15] J. Peng and R. J. Williams, 'Efficient learning and planning within the dyna framework', *Adaptive Behavior*, **1**(4), 437–454, (1993).

[16] Raquel Ros, Josep Lluis Arcos, Ramon López de Mántaras, and Manuela Veloso, 'A case-based approach for coordinated action selection in robot soccer', *Artificial Intelligence*, **173**(9-10), 1014–1039, (2009).

[17] G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems, 1994. Technical Report CUED/F-INFENG/TR 166. Cambridge University, Engineering Department.

[18] Vishal Soni and Satinder Singh, 'Using homomorphisms to transfer options across continuous reinforcement learning domains', in *Proceedings of the 21st National Conference on Artificial intelligence - Volume 1*, pp. 494–499. AAAI Press, (2006).

[19] Murray R. Spiegel, *Statistics*, McGraw-Hill, 1998.

[20] Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu, 'Keepaway soccer: From machine learning testbed to benchmark', in *RoboCup-2005: Robot Soccer World Cup IX*, eds., Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, volume 4020 of *Lecture Notes in Artificial Intelligence*, 93–105, Springer Verlag, Berlin, (2006).

[21] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann, 'Reinforcement Learning for RoboCup-soccer Keepaway', *Adaptive Behavior*, **13**(3), 165–188, (2005).

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[23] Richard S. Sutton, 'Integrated architectures for learning, planning and reacting based on approximating dynamic programming', in *Proceedings of the 7th International Conference on Machine Learning*, Austin, TX, (1990). Morgan Kaufmann.

[24] Matthew E. Taylor, Nicholas K. Jong, and Peter Stone, 'Transferring instances for model-based reinforcement learning', in *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Artificial Intelligence*, pp. 488–505, 2008.

[25] Matthew E. Taylor and Peter Stone, 'Transfer Learning for Reinforcement Learning domains: A survey', *Journal of Machine Learning Research*, **10**(1), 1633–1685, (2009).

[26] Matthew Edmund Taylor, *Autonomous inter-task transfer in reinforcement learning domains*, Ph.D. dissertation, University of Texas at Austin, Austin, TX, USA, 2008.

[27] E. L. Thorndike and R. S. Woodworth, 'The influence of improvement in one mental function upon the efficiency of other functions', *Psychological Review*, **8**, 247–261, (1901).

[28] Lisa Torrey, Trevor Walker, Jude W. Shavlik, and Richard Maclin, 'Using advice to transfer knowledge acquired in one Reinforcement Learning task to another', in *ECML*, eds., João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge, and Luís Torgo, volume 3720 of *Lecture Notes in Computer Science*, pp. 412–424. Springer, (2005).

[29] Andreas von Hessling and Ashok K. Goel, 'Abstracting reusable cases from reinforcement learning', in *6th International Conference on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Workshop Proceedings*, ed., Stefanie Brüninghaus, pp. 227–236, (2005).

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