Using Transfer Learning to Speed-Up Reinforcement Learning: a Cased-Based Approach

Luiz A. Celiberto Jr. and Jackson P. Matsuura
Technological Institute of Aeronautics, Brazil.
celibertojr@gmail.com, jackson@ita.br

Ramón López de Màntaras
Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain.
mantaras@iiia.csic.es

Reinaldo A. C. Bianchi
Dept. of Electrical Engineering, Centro Universitário da FEI, Brazil.
rbianchi@fei.edu.br

***Abstract*— Reinforcement Learning (RL) is a well known technique for the solution of problems where agents need to act with success in an unknown environment, learning through trial and error. However, this technique is not efficient enough to be used in applications with real world demands due to the time that the agent needs to learn.**

This paper investigates the use of Transfer Learning (TL) between agents to speed up the well known Q-learning Reinforcement Learning algorithm. The new approach presented here allows the use of cases in a case base as heuristics to speed up the Q-learning algorithm, combining Case-Based Reasoning (CBR) and Heuristically Accelerated Reinforcement Learning (HARL) techniques.

A set of empirical evaluations were conducted in the Mountain Car Problem Domain, where the actions learned during the solution of the 2D version of the problem can be used to speed up the learning of the policies for its 3D version.

The experiments were made comparing the Q-learning Reinforcement Learning algorithm, the HAQL Heuristic Accelerated Reinforcement Learning (HARL) algorithm and the TL-HAQL algorithm, proposed here. The results show that the use of a case-base for transfer learning can lead to a significant improvement in the performance of the agent, making it learn faster than using either RL or HARL methods alone.

I. Introduction

Reinforcement Learning (RL) is a very successful Artificial Intelligence sub-area [1]. It is concerned with the problem of learning from interaction to achieve a goal. Given an autonomous agent interacting with its environment via perception and action, on each interaction step the agent senses the current state s of the environment, and chooses an action a to perform. The action a alters the state s of the environment, and a scalar reinforcement signal r (a reward or penalty) is provided to the agent to indicate the desirability of the resulting state. The policy π is some function that tells the agent which actions should be chosen, and it is learned through trial-and-error interactions of the agent with its environment.

RL algorithms are very useful for solving a wide variety problems when the model is not known in advance, with many algorithms possessing guarantees of convergence to equilibrium [1], [2]. Unfortunately, the convergence of any RL algorithm may only be achieved after an extensive exploration of the state-action space, which is usually very time consuming.

One way to speed up the convergence of RL algorithms is by making use of a heuristic function in a manner similar to the use of heuristics in informed search algorithms. Heuristically Accelerated Reinforcement Learning (HARL) methods, which have been recently proposed [3], apply a conveniently chosen heuristic function for selecting the appropriate actions to perform in order to guide exploration during the learning process.

Although several methods have been successfully applied for speeding up RL algorithms by making use of heuristics, a very interesting option had not been explored yet: the transfer of learned policies, using a Case Based Reasoning approach to define an heuristic function. To address the learning transfer issue, this paper investigates the use of the heuristic function as a means to transfer 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.

Transfer learning is a paradigm of machine learning that reuses knowledge accumulate in a previous task to better learn a novel, but related, target task [4] and can be characterized as a gain or loss of proficiency in a task as a result of a practice in another task previously. Only recently the use of transfer learning has received great attention in learning agents and solving tasks with success compared to other techniques of machine learning [5].

The domain in which experiments were conducted in this work is the Mountain Car Problem Domain, where the actions learned during the solution of the 2D version of the problem can be used to speed up the learning of the policies for its 3D version. Nevertheless, the techniques described in this work is domain independent and can be used to solve a wide range of problems.

The paper is organized as follows: Section II briefly reviews the Reinforcement Learning problem and the section III describes the HARL approach to speed up RL and the HAQL algorithm. Section IV describes Case Based Reasoning. Section V describes the Transfer Learning problem and the section VI describes related work. Section VII describes the combination of the techniques and the modified formulation of the HAQL algorithm. Section VIII shows the experiments and the results and finally, section IX describes the conclusions of this work.

II. Reinforcement Learning and the Q–Learning algorithm

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 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.

The Q–learning algorithm [2] is a well-know RL technique that uses a strategy to learn an optimal policy π^* via learning of the action values. It iteratively approximates Q, provided the system can be modeled as an MDP, the reinforcement function is bounded, and actions are chosen so that every state-action pair is visited an infinite number of times. The Q–learning update rule is:

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

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 ($0 \leq \gamma < 1$); and α is the learning rate. To select an action to be executed, the Q–learning algorithm usually considers an $\epsilon - Greedy$ strategy:

$$\pi(s) = \begin{cases} \arg\max_a \hat{Q}(s, a) & \text{if } q \leq p, \\ a_{random} & \text{otherwise} \end{cases} \tag{2}$$

where:

- q is a random value uniformly distributed over $[0, 1]$ and p ($0 \leq p \leq 1$) is a parameter that defines the exploration/exploitation tradeoff: the larger p, the smaller is the probability of executing a random exploratory action.
- a_{random} is an action randomly chosen among those available in state s.

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.

III. Heuristically Accelerated Reinforcement Learning

A Heuristically Accelerated Reinforcement Learning (HARL) algorithm [3] 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) [3], as an extension of the Q–learning algorithm [2]. The only difference between the two algorithms is that in the HAQL makes use of an heuristic function $H(s, a)$ in the $\epsilon - greedy$ action choice rule, that can be written as:

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

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:

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

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 [3], together with the definition of an upper bound for the error in the estimation of Q. The complete HAQL algorithm is presented in Table I.

IV. Case Based Reasoning

Case Based Reasoning [6], [7] 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

TABLE I
THE HAQL 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 1.
    $s \leftarrow s'$.
  Until s is terminal.
Until some stopping criterion is reached.

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.

According to López de Mántaras *et al* [7], 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". Other steps that are usually found in CBR systems are the evaluation of the proposed solution, the revision of the solution, if required in light of its evaluation, and the retention (learning) of a new case, if the system has learned to solve a new 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:

$$case = (P, A).$$

The problem description P corresponds to the situation in which the case can be used. For example, for a Mountain Car problem the description of a case is the agent position x and its velocity ($\dot{x}$) . The solution description A is composed by the actions that must perform to solve the problem.

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 the quadratic distance between the problem and the case.

V. TRANSFER LEARNING

Transfer Learning is a paradigm of Machine Learning that reuses knowledge accumulate in a previous task to better learn a novel, but related, target task [4] and can be characterized as a gain of proficiency in a task as a result of a practice in another task previously. For example, the abilities acquired while learning to drive a car can be applied when one learns to drive a truck, making the second learning task easier. Although not yet comparable to the abilities of humans, transfer learning can be a very useful tool when faster learning is needed or when other learning techniques fail.

Transfer Learning is not a new idea: it has been studied in the psychological literature on transfer of learning since the work of Thorndike and Woodworth [8]. Also, TL has been used to transfer between machine learning tasks for some time now, as can be seen in the works of Caruana [9], [10], Thrun [11] or Thrun and Mitchell [12]. These works usually study transfer of learning in the context of classification, multitask learning and inductive learning.

According to Taylor [5], only recently the use of Transfer Learning for Reinforcement Learning has gained attention in the artificial intelligence community. In RL, the use of transfer learning reduces the search space of the agent, helping it to learn faster.

VI. RELATED WORK

Probably Drummond [13] was 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.

Some authors have been studying the transfer learning benefit to speed up reinforcement learning. For example Taylor, Stone, Liu and Littman [14] use inter-task mapping to directly transfer the action-value function to speed up learning. Stone and Liu [15] use knowledge learned in one task to improve learning another related task. Croonenborghs, Driessens and Bruynooghe [16] use a probability tree to learn a mapping automatically from interactions with the environment.

Other authors have been studying the use of RL together with CBR and the relation between them. Sharma *et al* [17] make use of CBR as a function approximator for RL, and RL as revision algorithm for CBR in a hybrid architecture system; Gabel and Riedmiller [18] also makes use of CBR in the task of approximating a function over high-dimensional, continuous spaces; Juell and Paulson [19] exploit the use of RL to learn similarity metrics in response to feedback from the environment; Auslander *et al* [20] use CBR to adapt quickly an RL agent to changing conditions of the environment by the use of previously stored policies and Li, Zonghai and Feng [21] propose an algorithm that makes use of knowledge acquired by reinforcement learning to construct and extend a case base. Finally, Bianchi, Ros and López de Mántaras [22] use CBR together with Heuristic Accelerated Reinforcement Learning to improve reinforcement learning by using case based heuristics.

Finally, some works on Transfer Learning have also combined CBR and RL, for example, van Hessing and Goel [23] describe a technique for abstracting reusable cases from RL, enabling the transfer of acquired knowledge to other instances of the same problem.

VII. COMBINING TRANSFER LEARNING WITH CASE BASED REASONING AND REINFORCEMENT LEARNING

Building on the model proposed by Ros [24], [25], each case consists of 3 parts: Description of the problem, containing all relevant information of the agent state; Solution to the

TABLE II
THE TL-HAQL 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
    If there is a case that can be reused:
      Compute $H_t(s,a)$ using Equation 4 with the actions suggested by the case selected.
      Select an action a using equation 3.
    If not:
      Select an action a using equation 2
    Execute the action a, observe $r(s,a)$, s'.
    Update the values of $Q(s,a)$ according to equation 1.
    $s \leftarrow s'$.
  Until s is terminal.
Until some stopping criterion is reached.

problem: action taken to resolve the problem, and finally the expected return for performing the action, which indicates the quality of the action stored in this case. Formally, the case can be described by a 3-tuples:

$$case = (P, A, R),$$

where P describes the problem A, the solution and R expected return.

To transfer the cases between two learning agents we propose the TL-HAQL (Transfer Learning Heuristically Accelerated Q–learning) algorithm, based in the CB-HAQL algorithm [22].

This algorithm works in two phases: the case base construction and the transfer of learning. In the first phase, the case base construction, the Q-learning algorithm is used to learn one task. After the learning stabilizes, ie., $\hat{Q}(s',a') - \hat{Q}(s,a)$ is close to zero, a case based is built with a predefined number of cases. In the second phase, the transfer of learning, the previously stored case base is used to accelerate the learning. During the learning of the second task, cases are retrieved, adapted to the current situation and a heuristic function that corresponds to the case is built and used. A case is retrieved if the similarity is above a certain threshold. After a case is retrieved, an heuristic is computed using Equation 4 and the action suggested by the case selected. If the case base does not contain a case that is similar to the current situation, the algorithm will behave as the traditional Q-learning algorithm. The TL-HAQL algorithm is presented in Table II.

The advantages of this proposal is that 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.

VIII. THE TRANSFER LEARNING EXPERIENCE

In this section we show how cases acquired in the 2D Mountain Car problem [26], can be transferred and used to speed up the learning in the 3D mountain car [27] problem.

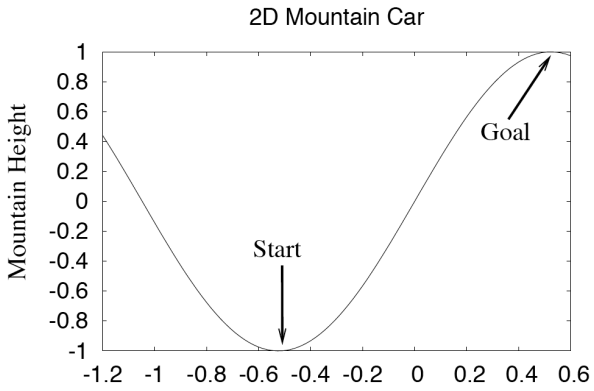

Fig. 1. The 2D Mountain Car Problem. (Image from [28]).

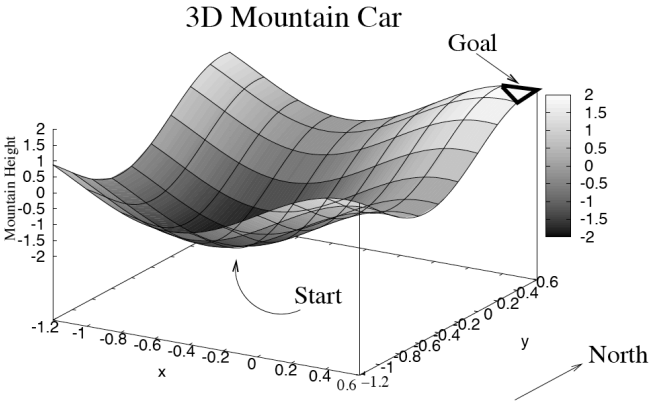

Fig. 2. The 3D Mountain Car Problem: the 3D task extends the Mountain Car problem into a surface (Image from [28]).

The Mountain Car Problem is a domain that has been traditionally used by researchers to test new reinforcement learning algorithms. In this problem, a car that is located at the bottom of a valley must be pushed back and forward until it reaches the top of a hill. The agent must generalize across continuous state variables in order to learn how to drive the car up to the goal state.

In 2D mountain car problem (Figure 1) two continuous variables describe the agents state: the horizontal position (x) restricted to the ranges [-1.2, 0.6] and velocity ($\dot{x}$) restricted to the ranges [-0.07, 0.07]. The agent may select one of three actions on every step: Left, Neutral, Right, which change the velocity by -0.0007, 0, and 0.0007 respectively.

The 3D mountain car (Figure 2) is similar to the 2D problem, extending it into a surface [28]. The agent starts at rest at the bottom of the hill and the goal is to reach the area in the upper right corner. The state is composed of four continuous state variables: x,$\dot{x}$,y,$\dot{y}$. The positions and velocities have ranges of [-1.2, 0.6] and [-0.07, 0.07], respectively. The agent can select from five actions at each timestep: Neutral, West, East, South, North. West and East modify $\dot{x}$ by -0.0007 and +0.0007 respectively, while South and North modify $\dot{y}$ by -0.0007 and +0.0007 respectively. The 3D problem is more difficult than the 2D because of the increased state space size and additional actions.

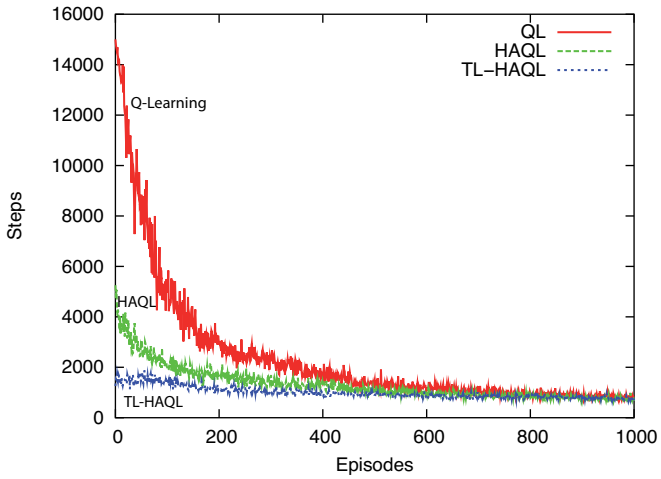

Fig. 3. The learning curves for the Q–learning, HAQL and TL-HAQL algorithms in the 3D Mountain Car Problem.

TABLE III
ACTION-MAPPING MOUNTAIN CAR

3D	2D
Neutral	Neutral
North	Right
East	Right
South	Left
West	Left

To learn the 3D task, the first algorithm to be evaluated is the Q–learning, described in section II, after that the HAQL, described in section III and finally the TL-HAQL algorithm proposed in section VII.

To build the case-base to be used by the TL-HAQL algorithm, the Q–learning algorithm is used in the 2D mountain car domain for 10.000 episodes (each episodes ends either after 15.000 steps or when the agent find the goal state). Acquiring cases begins when the learning stabilizes $(\hat{Q}(s', a') - \hat{Q}(s, a) \sim 0)$ which happens near the 9.000th episode. Each case contains the state (P), the action taken (A) and expected return (R), ie., the case was acquired in this format: P: -0.401242 -0.013694 A: 2 R: -1.0.

From the episode 9.000 and beyond, 500 cases are acquired by sampling the action-state set randomly. During this sampling, if a case contains the worst action for that state (i.e., the one with the lowest Q value), this case is discarded. In this way the case based is built using the best actions for the 2D problem. These cases will form the case base and will be transferred to the 3D mountain car problem.

When algorithm TL-HAQL selects a case, the action mapping defined in table III is used. Actions that accelerate the car towards the goal are mapped together and the same was made with the actions that accelerate the car away from the goal.

The heuristic used in the HAQL algorithm was defined using a simple rule: if the velocity is negative, chance course and go forward. The parameters used in the experiments were the same for all the algorithms: $\alpha = 0,9$, the exploration/ exploitation rate $= 0.2$, $\gamma = 0.9$ and $\eta = 1$. Values in the Q table were randomly initiated.

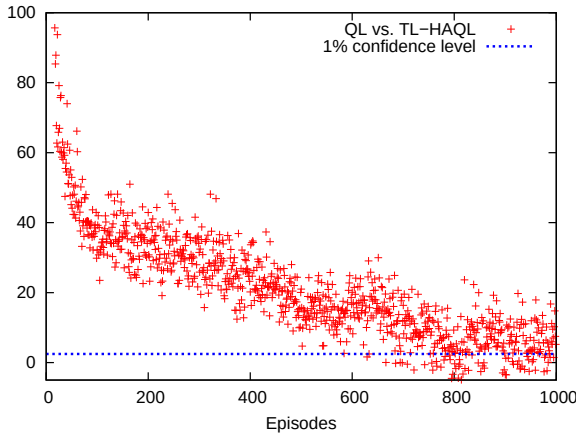

Fig. 4. Results from Student's t-test between Q-learning and TL-HAQL.

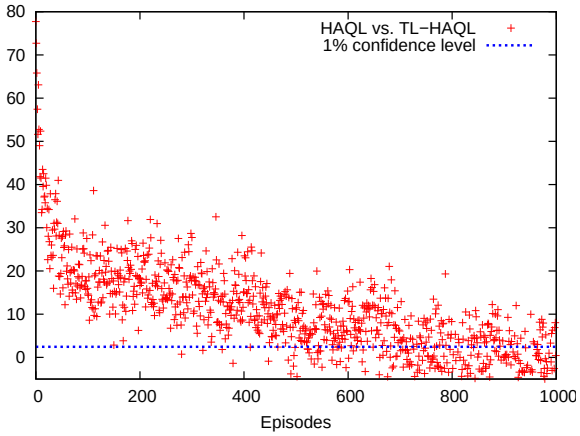

Fig. 5. Results from Student's t-test between HAQL and TL-HAQL.

Thirty training sessions were executed for the three algorithms, with each session consisting of 1000 episodes. Figure 3 show the learning curves for all algorithms. It can be see that the performance of the Q–learning and the HAQL are worse than that of the TL-HAQL at the initial learning phase; later the performance of the three algorithms become more similar, as expected.

Student's t-test [29] was used to verify the hypothesis that the transfer of learning speeds up the learning process. For the experiments the value of the module of T was computed for each episode using the same data presented in figure 3. The result, presented in figures 4 and 5, shows that TL-HAQL performs clearly better than Q–learning until the 700th episode, and HAQL until the 400th episode, with a level of confidence greater than 99%. After that, the results became closer.

IX. CONCLUSION

Transfer learning from one agent to another agent by means of the heuristic function speeds up the convergence of the algorithm when compared to a normal Q–learning

algorithm. The use of CB to transfer the learning is a important tool and make the TL-HAQL algorithm faster and very useful to speed-up reinforcement learning algorithms such as Q–learning. Regarding the use of the TL-HAQL algorithm to accelerate the RL, it is worth noticing that the agent converges to optimality faster because there is no need to explore the whole search space.

Future works include automating the mapping between the actions in the 2D and the 3D mountain car problems, so that the Table III can be built automatically, and experimenting with better ways to build the case base.

X. ACKNOWLEDGMENTS

Luiz Celiberto acknowledge the support of CNPq (Grant No. xxxx) and Reinaldo Bianchi acknowledge the support of the CNPq (Grants No. 201591/2007-3). This work has been partially funded by the 2009-SGR-1434 grant of the Generalitat de Catalunya, the NEXT-CBR project, and FEDER funds.

REFERENCES

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

[2] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.

[3] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa, "Accelerating autonomous learning by using heuristic selection of actions," *Journal of Heuristics*, vol. 14, no. 2, pp. 135–168, 2008.

[4] M. E. Taylor, N. K. Jong, and P. Stone, "Transferring instances for model-based reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Artificial Intelligence, vol. 5212, September 2008, pp. 488–505.

[5] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. 1, pp. 1633–1685, 2009.

[6] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[7] R. L. de Mántaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson, "Retrieval, reuse, revision and retention in case-based reasoning," *Knowl. Eng. Rev.*, vol. 20, no. 3, pp. 215–240, 2005.

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

[9] R. Caruana, "Learning many related tasks at the same time with backpropagation," in *In Advances in Neural Information Processing Systems 7*. Morgan Kaufmann, 1995, pp. 657–664.

[10] ——, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[11] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Advances in Neural Information Processing Systems*. The MIT Press, 1996, pp. 640–646.

[12] S. Thrun and T. M. Mitchell, "Learning one more thing," in *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1217–1223.

[13] C. Drummond, "Accelerating reinforcement learning by composing solutions of automatically identified subtasks," *Journal of Artificial Intelligence Research*, vol. 16, pp. 59–104, 2002.

[14] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via intertask mappings for temporal difference learning," *Journal of Machine Learning Research*, vol. 8, no. 1, pp. 2125–2167, 2007.

[15] Y. Liu and P. Stone, "Value-function-based transfer for reinforcement learning using structure mapping," in *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, July 2006, pp. 415–20.

[16] T. Croonenborghs, K. Driessens, and M. Bruynooghe, "Learning relational options for inductive transfer in relational reinforcement learning," in *ILP'07: Proceedings of the 17th international conference on Inductive logic programming*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 88–97.

[17] M. Sharma, M. Holmes, J. C. Santamaría, A. Irani, C. L. I. Jr., and A. Ram, "Transfer learning in real-time strategy games using hybrid cbr/rl," in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 1041–1046.

[18] T. Gabel and M. A. Riedmiller, "CBR for state value function approximation in reinforcement learning," in *6th International Conference on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Proceedings*, ser. Lecture Notes in Computer Science, H. Muñoz-Avila and F. Ricci, Eds., vol. 3620. Springer, 2005, pp. 206–221.

[19] P. Juell and P. Paulson, "Using reinforcement learning for similarity assessment in case-based systems," *IEEE Intelligent Systems*, vol. 18, no. 4, pp. 60–67, 2003.

[20] B. Auslander, S. Lee-Urban, C. Hogg, and H. Muñoz-Avila, "Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning," in *ECCBR*, ser. Lecture Notes in Computer Science, K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, Eds., vol. 5239. Springer, 2008, pp. 59–73.

[21] Y. Li, C. Zonghai, and C. Feng, "A case-based reinforcement learning for probe robot path planning," in *4th World Congress on Intelligent Control and Automation, Shanghai, China*, 2002, pp. 1161– 1165.

[22] R. A. C. Bianchi, R. Ros, and R. L. 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*, ser. Lecture Notes in Computer Science, L. McGinty and D. C. Wilson, Eds., vol. 5650. Springer, 2009, pp. 75–89.

[23] A. von Hessling and A. 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*, S. Brüninghaus, Ed., 2005, pp. 227–236.

[24] R. Ros, "Action selection in cooperative robot soccer using case-based reasoning," Ph.D. dissertation, Universitat Autònoma de Barcelona, Barcelona, 2008.

[25] R. Ros, J. L. Arcos, R. L. de Mántaras, and M. Veloso, "A case-based approach for coordinated action selection in robot soccer," *Artificial Intelligence*, vol. 173, no. 9-10, pp. 1014–1039, 2009.

[26] A. Moore, "Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces," in *Machine Learning: Proceedings of the Eighth International Conference*, L. Birnbaum and G. Collins, Eds. 340 Pine Street, 6th Fl., San Francisco, CA 94104: Morgan Kaufmann, June 1991.

[27] M. E. Taylor, G. Kuhlmann, and P. Stone, "Autonomous transfer for reinforcement learning," in *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2008.

[28] M. E. Taylor, "Autonomous inter-task transfer in reinforcement learning domains," Ph.D. dissertation, University of Texas at Austin, Austin, TX, USA, 2008.

[29] M. R. Spiegel, *Statistics*. McGraw-Hill, 1998.