ECAI 2012
Luc De Raedt et al. (Eds.)

© 2012 The Author(s).
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-098-7-169

Heuristically Accelerated Reinforcement Learning: Theoretical and Experimental Results

Reinaldo A. C. Bianchi[1], **Carlos H. C. Ribeiro**[2] and **Anna H. R. Costa**[3]

Abstract. Since finding control policies using Reinforcement Learning (RL) can be very time consuming, in recent years several authors have investigated how to speed up RL algorithms by making improved action selections based on heuristics. In this work we present new theoretical results – convergence and a superior limit for value estimation errors – for the class that encompasses all heuristics-based algorithms, called Heuristically Accelerated Reinforcement Learning. We also expand this new class by proposing three new algorithms, the Heuristically Accelerated Q(λ), SARSA(λ) and TD(λ), the first algorithms that uses both heuristics and eligibility traces. Empirical evaluations were conducted in traditional control problems and results show that using heuristics significantly enhances the performance of the learning process.

1 Introduction

One of the main problems of Reinforcement Learning (RL) [12] algorithms is that they typically suffer from very slow learning rates, requiring a huge number of iterations to converge to a good solution. This problem gets worse in tasks with high dimensional or continuous state spaces and when the learner receives sparse rewards.

A way to speed up RL algorithms is by making use of a conveniently chosen heuristic function, which is used for selecting appropriate actions to perform in order to guide state space exploration during the learning process. Several methods have successfully considered a heuristic function in RL, including the use of prior domain knowledge to infer a heuristics [5]; the use of a previous problem solution as heuristics in the initialization of a Q-table [7]; the use of information from the learning process to infer a heuristics in execution time [3, 4] and the reuse of previously learned policies, using a Case-Based Reasoning approach [6].

In this work we present new theoretical results – convergence and a superior limit for value estimation errors – for the class that encompasses all heuristics-based algorithms, called Heuristically Accelerated Reinforcement Learning. We also expand this class by proposing three new algorithms, the Heuristically Accelerated Q(λ), the HA-SARSA(λ) and the HA-TD(λ), the first algorithms that use both heuristics and eligibility traces. Experiments for this work were conducted in two traditional control domains: the Mountain Car Problem and the Cart-Pole Problem [12], using function approximators as both domains use continuous state spaces. Nevertheless, the technique described herein is domain-independent and can be used to solve a wide range of problems.

[1] Centro Universitário FEI, Brazil, email: rbianchi@fei.edu.br
[2] Instituto Tecnológico de Aeronáutica, Brazil, email: carlos@ita.br
[3] Escola Politécnica da Universidade de São Paulo, Brazil, email: anna.reali@poli.usp.br

This paper is organized as follows: Sections 2 and 3 briefly reviews RL and the heuristic approach to speed up RL. Section 4 presents new theoretical results for the HARL algorithm and Section 5 proposes three new algorithms. Section 6 describes results obtained by the use of heuristic functions in conjunction with some classic RL algorithms in benchmark problems. Finally, Section 7 provides the conclusions and indicates avenues by which the research proposed in this paper can be extended.

2 Reinforcement Learning

Let us consider a single agent interacting with its environment via perception and action. On each interaction step t, the agent senses the current state s_t of the environment, and chooses an action a_t to perform. The action a_t alters the state s_t into a new state s_{t+1}, and a scalar reinforcement signal r_t (a reward or penalty) is provided to the agent to indicate the desirability of the resulting state.

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 possible actions; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$: is a state transition function, where $T(s_t, a_t, s_{t+1})$ is the probability that performing action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ at time t will lead to state $s_{t+1} \in \mathcal{S}$ at time $t + 1$; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: is a finite set of bounded reinforcements (payoffs), $r(\mathbf{s_t}, a_t) \in \mathbb{R}$.

The goal of the agent in the most common formulation of the RL problem is to learn an optimal policy of actions, π^*, which maximizes the expected discounted value function [12, Equation 3.8]:

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \quad (1)$$

for any starting state s, when $\mathcal{R}$ and $\mathcal{T}$ are not known.

Identified by Sutton and Barto [12] as "the central and novel idea of reinforcement learning", the temporal-difference (TD) learning is the simplest method for learning the value function. It estimates the expected discounted value function using:

$$\hat{V}_{t+1}(s_t) \leftarrow \hat{V}_t(s_t) + \alpha \left[r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}_t(s_t) \right]. \quad (2)$$

Another strategy to learn an optimal policy π^* is to allow the agent to learn the action-value function. $Q^{\pi}(s, a)$ is defined as [12, Equation 3.9]:

$$Q^{\pi}(s, a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \quad (3)$$

which represents the expected return for taking action a when visiting state s and following policy π thereafter.

Two algorithms that can be used to iteratively approximate Q are the Q-learning [15] and the SARSA [14] algorithms. The Q learning rule is:

$$\hat{Q}_{t+1}(s_t,a_t) \leftarrow \hat{Q}_t(s_t,a_t) + \alpha\left[r(s_t,a_t) + \gamma \max_{a_{t+1}} \hat{Q}_t(s_{t+1},a_{t+1}) - \hat{Q}_t(s_t,a_t)\right] \quad (4)$$

where γ is a discount factor and α is the learning rate.

The SARSA algorithm is a modification of Q-learning that eliminates the maximization of the actions in equation (4), separating the choice of the actions to be taken from the update of the Q values. The Q(λ) [15] and the SARSA(λ) [10] algorithms extend the original algorithms by, instead of updating a state-action pair at each iteration, updating all pairs in a eligibility trace, proposed initially in the $TD(\lambda)$ algorithm.

Finally, to work with problems with continuous state spaces, algorithms can be implemented using function approximators – instead of a table – to compute the action-value function. In this work the algorithms used uses two function approximators to compute the Q value: a CMAC function approximator [1] and the function approximator described in Barto *et al.* [2].

3 Heuristics in Reinforcement Learning

Bianchi *et al.* [4] defined a Heuristically Accelerated Reinforcement Learning (HARL) algorithm as a way to solve an MDP problem with explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ for influencing the choice of actions by the learning agent. 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 heuristic function is an action policy modifier which does not interfere with the standard bootstrap-like update mechanism of RL algorithms. In the HARL algorithms, instead of using only the value (or action-value) estimation in the action selection method of an RL algorithm, a mathematical combination between the estimation function and a heuristic function is used:

$$\left[\mathsf{F}_t(s_t,a_t) \bowtie \xi H_t(s_t,a_t)^{\beta}\right] \quad (5)$$

where: $\mathcal{F} : \mathcal{S}\times\mathcal{A} \rightarrow \mathbb{R}$ is an estimate of a value function that defines the expected cumulative reward (for example, it is $\hat{Q}_t(s_t,a_t)$ for the Q-learning); $\mathcal{H} : \mathcal{S}\times\mathcal{A} \rightarrow \mathbb{R}$ is the heuristic function that plays a role in the action choice, defining the importance of executing action a_t in state s_t; $\bowtie$ is a function that operates on real numbers and produces a value from an ordered set and ξ and β are design parameters used to control the influence of the heuristic function (they can be lowered to decrease the influence of the heuristic with time).

This formulation is more general than other previous proposals, allowing heuristics to be used with different action selection methods and RL algorithms. One proposed strategy for action choice is an $\epsilon - Greedy$ mechanism where $H_t(s_t,a_t)$ is considered, thus:

$$\pi(s_t) = \begin{cases} \pi'(s_t) & \text{if } q \leq p, \\ a_{random} & \text{otherwise} \end{cases} \quad (6)$$

where:

$$\pi'(s_t) = \arg\max_{a_t}\left[\mathsf{F}_t(s_t,a_t) \bowtie \xi H_t(s_t,a_t)^{\beta}\right], \quad (7)$$

p is a parameter that define the exploration/exploitation tradeoff, q is a random number between 0 and 1 and a_{random} is an action randomly chosen among those available in state s_t.

Another possible strategy that can use heuristics is Boltzmann exploration [12], a strategy that assigns a probability to any possible action according to its expected utility, i.e., actions with higher Q have greater probability of being chosen. A HARL using this strategy chooses action a with probability:

$$Pr(a_t|s_t) = \frac{e^{[\mathsf{F}_t(s_t,a_t) \bowtie \xi H_t(s_t,a_t)^{\beta}]/\tau}}{\sum_{a' \in A} e^{[\mathsf{F}_t(s_t,a') \bowtie \xi H_t(s_t,a')^{\beta}]/\tau}} \quad (8)$$

where τ is the temperature, which decreases with time.

In general, the value of $H_t(s_t,a_t)$ must be larger than the variation among the values of $\mathsf{F}(s_t,a)$ for a given $s_t \in S$, so that it can influence the action choice. On the other hand, it must be as small as possible in order to minimize the error. If $\bowtie$ is a sum and $\xi = \beta = 1$, a heuristics that can be used with the $\epsilon - Greedy$ mechanism can be defined by:

$$H_t(s_t,a_t) = \begin{cases} \max_{a\in\mathcal{A}}\left[\mathsf{F}_t(s_t,a)\right] - \\ \mathsf{F}_t(s_t,\pi^H(s_t)) + \eta & \text{if } a_t = \pi^H(s_t), \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

where η is a small value and $\pi^H(s_t)$ is a heuristics obtained using an appropriate method, that is desired to be used in s_t.

For instance, let [1.0 1.1 1.2] be the values of $\mathsf{F}(s_t,a)$ for three possible actions [a_1 a_2 a_3] for a given state s_t. If the desired action is the first one (a_1), we can use $\eta = 0.01$, resulting in $H(s_t,a_1) = 0.21$ and zero for the other actions.

An important characteristic of a HARL algorithm is that the heuristic function can be modified or adapted online, as learning progresses and new information for enhancement of the heuristics becomes available. In particular, either prior domain information or initial learning stage information can be used to define heuristics to accelerate learning.

4 Theoretical results

As the heuristic function is used only in the choice of the action to be taken, a new HARL algorithm is different from the original RL one only in the way exploration is carried out. As the RL algorithm operation is not modified, many of the conclusions reached in RL are also valid for HARL. In this section we present new theorems that confirm this statement and limit the maximum error caused by using a heuristics.

Theorem 1 *Consider a HARL agent learning in a deterministic MDP, with finite sets of states and actions, bounded rewards* ($\exists c \in \mathbb{R}; (\forall s,a), |r(s,a)| < c$)*, discount factor* γ *such that* $0 \leq \gamma < 1$ *and where the values used on the heuristic function are bounded by* $(\forall s_t,a_t)\ h_{min} \leq H(s_t,a_t) \leq h_{max}$*. For this agent, the* F_t *values will converge to* F^**, with probability one uniformly over all the states* $s \in \mathcal{S}$*, if each state-action pair is visited infinitely often.*

Proof: In HARL, the update of the value function approximation does not depend explicitly on the value of the heuristics. The necessary conditions for the convergence of an RL algorithm that could be affected with the use of the heuristics, are the ones that depend on the choice of the action. Of the conditions presented in [8], the only one that depends on the action choice is the necessity of infinite visitation to each pair state-action. As equation 6 considers an exploration strategy ϵ– *greedy* regardless of the fact that the value function is influenced by the heuristics, the infinite visitation condition

is guaranteed and the algorithm converges. The condition of infinite visitation of each state-action pair can be considered valid for other exploration strategies (e.g., Boltzmann exploration in Equation 8) by using other visitation strategies, such as intercalating steps where the algorithm makes alternate use of the heuristics and exploration steps, receding the influence of the heuristics with time or using the heuristics during a period of time, smaller than the total learning time for Q–learning. $q.e.d.$

The following theorem guarantees that small errors in the approximation of an optimal value function cannot produce arbitrarily bad performance when actions are selected using the ϵ-greedy rule influenced by heuristics (Equation 6). The proofs here are based on the work of Singh [11, Section 4.5.1].

Definition 1 *The loss in the approximation of the value function caused by the use of heuristics can be defined as:*

$$L_H(s_t) = \mathsf{F}_t(s_t, \pi^*) - \mathsf{F}_t(s_t, \pi^H), \forall s_t \in S, \tag{10}$$

where $\mathsf{F}_t(s_t, \pi^H)$ is the estimated value function calculated for the policy indicated by the heuristics, π^H.

The theorem presented below defines the upper bound for the loss $L_H(s_t), \forall s_t \in S$.

Theorem 2 *The maximal loss that can be caused by the use of a heuristic function bounded by $h_{min} \leq H(s_t, a_t) \leq h_{max}$ in a HARL algorithm learning in a deterministic MDP, with finite sets of states and actions, bounded rewards $(\forall s_t, a_t)\ r_{min} \leq r(s_t, a_t) \leq r_{max}$, discount factor γ such that $0 \leq \gamma < 1$ and where $\bowtie$ is the addition, has an upper bound:*

$$L_H(s_t) \leq \xi\ [h^{\beta}_{max} - h^{\beta}_{min}]. \tag{11}$$

Proof: There exists a state z that causes maximum loss: $\exists z \in S, \forall s \in S, L_H(z) \geq L_H(s)$. For this state z, consider an optimal action $a = \pi^*(z)$ and the action indicated by the heuristics $b = \pi^H(z)$. Using a results in the state x, and using b results in the state y. Because the choice of action is made following an ϵ-greedy policy, b must seem at least as good as a:

$$\mathsf{F}_t(z, a) + \xi H_t(z, a)^{\beta} \leq \mathsf{F}_t(z, b) + \xi H_t(z, b)^{\beta}.$$

Rearranging this equation we have:

$$\mathsf{F}_t(z, a) - \mathsf{F}_t(z, b) \leq \xi H_t(z, b)^{\beta} - \xi H_t(z, a)^{\beta}. \tag{12}$$

Using the definition of the loss in the approximation of the value function (Equation 10) and the definition of a and b:

$$L_H(z) = \mathsf{F}_t(z, a) - \mathsf{F}_t(z, b). \tag{13}$$

Substituting (12) in (13) gives:

$$L_H(z) \leq \xi \left[H_t(z, b)^{\beta} - H_t(z, a)^{\beta} \right]. \tag{14}$$

Because the action b is chosen instead of the action a, $H_t(z, b)^{\beta} \geq H_t(z, a)^{\beta}$. As the value of $\mathcal{H}$ is bounded by $h_{min} \leq H(s_t, a_t) \leq h_{max}$, it can be concluded that:

$$L_H(s_t) \leq \xi\ [h^{\beta}_{max} - h^{\beta}_{min}], \forall s_t \in S. \quad q.e.d. \tag{15}$$

Is it possible to improve the definition of the maximal loss. The following two lemmas are results known to be valid for RL algorithms, which are also valid for the HARL algorithms.

Lemma 1 *For any RL or HARL algorithm, learning in a deterministic MDP, with finite sets of states and actions, bounded rewards $(\forall s_t, a_t)\ r_{min} \leq r(s_t, a_t) \leq r_{max}$, discount factor γ such that $0 \leq \gamma < 1$, the maximum value that $\mathsf{F}(s_t, a_t)$ can reach has an upper bound of $r_{max}/(1-\gamma)$.*

Proof: From the expected discounted value function definition (Equation 1) we have:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \tag{16}$$

And from the definition ot the action-value function (Equation 3):

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= r_t + \gamma V^{\pi}(s_{t+1}) \\ &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \\ &= \sum_{i=0}^{\infty} \gamma^i r_{t+i}. \end{aligned}$$

Therefore,

$$\mathsf{F}_t(s_t, a_t) = Q^{\pi}(s_t, a_t) = V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

and

$$\mathsf{F}_t(s_t, a_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}, \tag{17}$$

were r_{t+i} is the sequence of rewards obtained when starting from s_t, using π to select the actions and where γ is the discount factor such that $0 \leq \gamma < 1$.

Assuming that, in the best case, all received rewards in all steps were $r_{t+i} = r_{max}$, we have that:

$$\begin{aligned} \max \mathsf{F}(s_t, a_t) &= r_{max} + \gamma r_{max} + \gamma^2 r_{max} + \ldots + \gamma^n r_{max} \\ &= \sum_{i=0}^{n} \gamma^i r_{max} \end{aligned}$$

Finally, in the limit $n \to \infty$, we have:

$$\begin{aligned} \max \mathsf{F}(s_t, a_t) &= \lim_{n \to \infty} \sum_{i=0}^{n} \gamma^i r_{max} \\ &= \frac{r_{max}}{1-\gamma} \quad \square \end{aligned}$$

If the positive reward is given only when the terminal state is reached, $r_t \leq r_{max}$ there are no other rewards for $t \geq t+1$, we conclude that $\forall (s_t, a_t), \max \mathsf{F}(s_t, a_t) \leq r_{max}$.

Lemma 2 *For any RL or HARL algorithm learning in a deterministic MDP, with finite sets of states and actions, bounded rewards $(\forall s_t, a_t)\ r_{min} \leq r(s_t, a_t) \leq r_{max}$, discount factor γ such that $0 \leq \gamma < 1$, the minimum value that $\mathsf{F}(s_t, a_t)$ can reach has a lower bound of $r_{min}/(1-\gamma)$.*

Proof: Assuming that, in the worst case, all received rewards in all steps were $r_{t+i} = r_{min}$, we have that:

$$\begin{aligned} \min \mathsf{F}(s_t, a_t) &= r_{min} + \gamma r_{min} + \gamma^2 r_{min} + \ldots + \gamma^n r_{min} \\ &= \sum_{i=0}^{n} \gamma^i r_{min} \end{aligned}$$

In the limit when $n \to \infty$:

$$\begin{aligned} \min \mathsf{F}(s_t, a_t) &= \lim_{n \to \infty} \sum_{i=0}^{n} \gamma^i r_{min} \\ &= \frac{r_{min}}{1-\gamma} \quad \square \end{aligned}$$

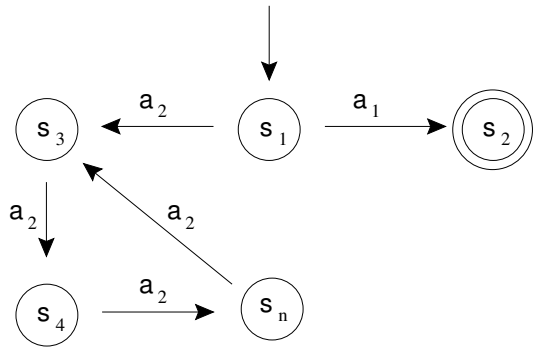

Figure 1. Problem where the state s_1 have actions that will receive both the maximum and minimum values for the action-value function $\mathsf{F}(s_t, a_t)$.

Theorem 3 *The maximal loss that can be caused by the use of a heuristic function in a HARL algorithm learning in a deterministic MDP, with finite sets of states and actions, bounded rewards*($\forall s_t, a_t$) $r_{min} \leq r(s_t, a_t) \leq r_{max}$*, discount factor* γ *such that* $0 \leq \gamma < 1$ *and where* $\bowtie$ *is the addition, has an upper bound:*

$$L_H(s_t) = \xi \left[\frac{r_{max} - r_{min}}{1-\gamma} + \eta \right]^{\beta}. \quad (18)$$

Proof: From Equation 9, we have:

$$h_{min} = 0 \text{ if } a_t \neq \pi^H(s_t), \text{ and}$$
$$h_{max} = \max_{a \in \mathcal{A}} [\mathsf{F}_t(s_t, a)] - \mathsf{F}_t(s_t, \pi^H(s_t)) + \eta \text{ if } a_t = \pi^H(s_t). \quad (19)$$

The value of the heuristics will be maximum when both the $\max \mathsf{F}(s_t, a_t)$ as the $\min \mathsf{F}(s_t, a_t)$, $\forall s_t \in S$, $a_t \in A$ are found in the same state s_t. In this case

$$h_{max} = \frac{r_{max}}{1-\gamma} - \frac{r_{min}}{1-\gamma} + \eta. \quad (20)$$

By substitution of h_{max} e h_{min} in the result of Theorem 2, we have:

$$\begin{aligned} L_H(s_t) &= \xi \left[h_{max}^{\beta} - h_{min}^{\beta} \right] \\ &= \xi \left[\left(\frac{r_{max}}{1-\gamma} - \frac{r_{min}}{1-\gamma} + \eta \right)^{\beta} - 0^{\beta} \right] \\ &= \xi \left[\frac{r_{max} - r_{min}}{1-\gamma} + \eta \right]^{\beta}. \quad q.e.d. \end{aligned}$$

Figure 1 shows an example of problem configuration where both the $\max \mathsf{F}(s_t, a_t)$ and the $\min \mathsf{F}(s_t, a_t)$ are found in the same state, s_1. In it, state s_2 is a terminal state; move to s_2 generates a reward r_{max} and any other movement generates a reward r_{min}.

5 HARL Algorithms

A generic procedure for a HARL algorithm was defined by Bianchi *et al.* [4] as a process that is sequentially repeated until a stopping criteria is met (Algorithm 1). Based on this description it is possible to create many new algorithms from existing RL ones.

The first HARL algorithm proposed was the Heuristically Accelerated Q–Learning (HAQL) [3], as an extension of the Q–Learning algorithm [15]. The only difference between the two algorithms is that

Algorithm 1 The HARL generic algorithm [Bianchi *et al.* 2008]

Produce an arbitrary estimation for the value function.
Define an initial heuristic function $H_0(\cdot, \cdot)$.
Observe the current state s_t.
repeat
  Select an action a_t by adequately combining the heuristic function and the value function.
  Execute a_t.
  Receive the reinforcement $r(s_t, a_t)$ and observe s_{t+1}.
  Update value (or the action-value) function.
  Update $H_t(s_t, a_t)$ using an appropriate method.
  Update state $s_t \leftarrow s_{t+1}$
until until a stopping criteria is met

HAQL makes use of a heuristic function $H(s, a)$ in the $\epsilon - greedy$ action choice rule, that can be written as:

$$\pi(s) = \begin{cases} \arg\max_a \left[\hat{Q}(s, a) + \xi H(s, a)^{\beta} \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \quad (21)$$

In this work we propose three new algorithms: the HA-Q(λ), which extends the Q(λ) algorithm by using the same action choice rule as the HAQL (Eq. 21), the HA-SARSA(λ), which extends the SARSA(λ) algorithm [10] in the same way, and the HA-TD(λ), that extends the traditional TD(λ) algorithm [13], using an action choice rule in which a probability function is influenced by the heuristic (shown in section 6.2). Except for the new action choice rule, the new algorithms works exactly as the original ones.

6 Experiments using Heuristics

This section presents two experiments conducted to verify that the approach based on heuristics can be applied to different RL algorithms, that it is domain independent and that it can be used in problems with continuous state space. The heuristics used were defined based on *a priori* knowledge of the domain. It is important to notice that the heuristics used here are not a complete solution (i.e., the optimal policy) to solve the problems.

6.1 The Mountain Car problem using HAQL and HA-SARSA(λ)

The Mountain Car Problem [9] 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.

Two continuous variables describe the car state: the horizontal position x restricted to the ranges [-1.2, 0.6] and velocity $v = \dot{x}$ restricted to the ranges [-0.07, 0.07]. The car may select one of three actions on every step: Left ($F = -1$), Neutral ($F = 0$) , Right ($F = 1$) , which change the velocity by -0.0007, 0, and 0.0007, respectively.

To solve this problem, six algorithms were used: the Q–Learning, the SARSA(λ), the Q(λ), the HAQL, the HA-SARSA(λ) and the HA-Q(λ), the first three, classic RL algorithms, and the last three, heuristic versions of them. Because the input variables are continuous, a CMAC function approximator [1] with 10 layers and 8 input positions for each variable was used to represent the value-action function (in the six algorithms).

The heuristics used was defined following a simple rule: always increase the module of the velocity $|\dot{x}|$. The value of the heuristics used in the HARLs is defined using Eq. (9) as: $H(x_t, v_t, F_t) =$

$$\begin{cases} \max\limits_{a \in \mathcal{A}} \hat{Q}(x_t, v_t, a) - \\ \hat{Q}(x_t, v_t, F_t) + \eta \\ \\ 0 \end{cases} \text{if} \begin{cases} v_t > 0 \text{ and } F_t = +1. \\ \textit{or} \\ v_t < 0 \text{ and } F_t = -1. \end{cases} \quad (22)$$
$$\text{otherwise.}$$

The parameters used in the simulation are the same for all algorithms: $\alpha = 0,5$, $\gamma = 0,99$, $\lambda = 0,9$, exploration rate = 10%, $\eta = 10$. The reward is -10 when applying a force ($F = -1$ or $F = 1$), -1 when $F = 0$, and 1000 when reaching the goal state.

Table 1 shows, for the six algorithms, the number of steps of the best solution and the time to find it (average of 30 training sessions limited to 500 episodes). It may be noted that the Q–Learning algorithm has the worst performance, as expected. It can also be seen that the algorithms that use heuristics are faster than the others.

Algorithm	Best Solution (steps)	Time (in sec.)
Q-Learning	430 ± 45	31 ± 3
SARSA(λ)	171 ± 14	41 ± 18
Q(λ)	123 ± 11	24 ± 14
HAQL	115 ± 1	7 ± 5
HA-SARSA(λ)	119 ± 1	4 ± 1
HA-Q(λ)	107 ± 1	4 ± 1

Table 1. Results for the Mountain Car problem: average number of steps of the best solution and the average time to find it.

Figure 2 shows the evolution of the number of steps needed to reach the goal for the six algorithms (average of 30 training sessions). As expected, the Q–learning has the worst performance (the beginning of its evolution is not presented because values are above $2,000$ steps) and the HARL algorithms present the best results. As the learning proceeds, the performance of all algorithms become similar, as expected (Q-learning will reach the optimal solution after $20,000$ steps). This figure also allows one to infer the reason why the time required for the RL algorithms to find the solution with fewer steps (presented in table 1) is greater than the time needed by the HARL algorithms: the smaller number of steps executed by the HARLs at the beginning of training.

The paths made by the car in the state space (position $\times$ speed) at the first training session, when controlled by the SARSA(λ) and HA-SARSA(λ) algorithms can be seen in Figure 3 (the optimal control policy is also presented). It can be seen how SARSA(λ) explores the environment at the beginning of training and, when compared with HA-SARSA(λ), one can notice that the great advantage of the HARL algorithms is to not perform such an intense exploration of the state space.

Finally, Student's t–test was used to verify the hypothesis that the use of heuristics speeds up the learning process. Results confirm the hypothesis, with a confidence level greater than 99%.

6.2 The cart-pole problem using HATD(λ)

The cart-pole task is used since the early work on RL, such as [2]. The goal of the cart-pole task is to maintain the vertical position of the pole while keeping the car within a fixed boundary [12]. A failure occurs when the pole is tilted more than 12 degrees from vertical or if

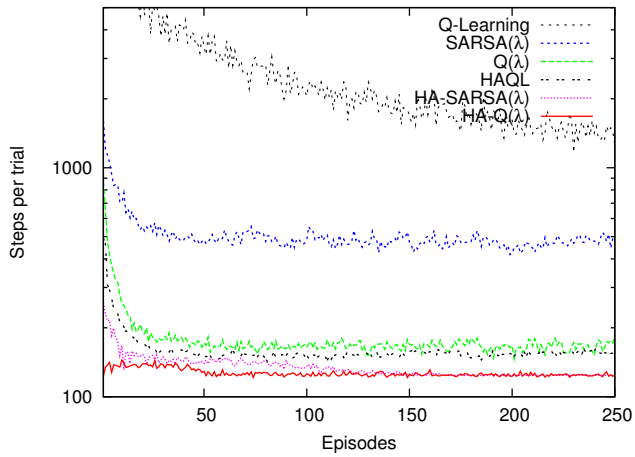

Figure 2. Evolution of the number of steps needed to reach the goal for the six algorithms (average of 30 training sessions, y axis in log scale).

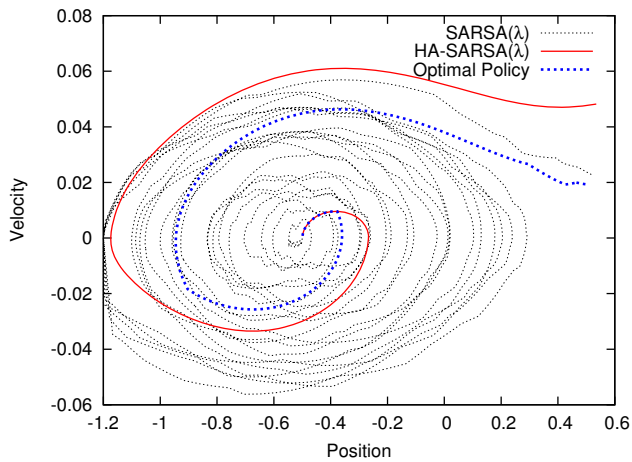

Figure 3. Paths made by the car in the state space.

the cart hits the end of the track. The state variables for this problem are continuous: the position of the cart $y_t \in [-2.4, 2.4]$, the speed of the cart $\dot{y}_t$, the angle between the pole and the vertical $\theta_t \in [-12, 12]$ degrees and the rate of change of the poles angle $\dot{\theta}_t$ (the dynamic equations can be found in [2]).

We use two algorithms to solve this problem: TD(λ) and HATD(λ), which implements a heuristic version of the TD(λ) algorithm. The heuristics used was similar to that of the previous section: if the pole is falling to the left, move the cart to the left, if it is falling to the right, move to the right:

$$H(y_t, \theta_t) = \begin{cases} +\eta & \text{if } \theta_t > 0, \forall y_t \\ -\eta & \text{if } \theta_t < 0, \forall y_t \end{cases} \quad (23)$$

This heuristics influences the choice of actions, which is given by the

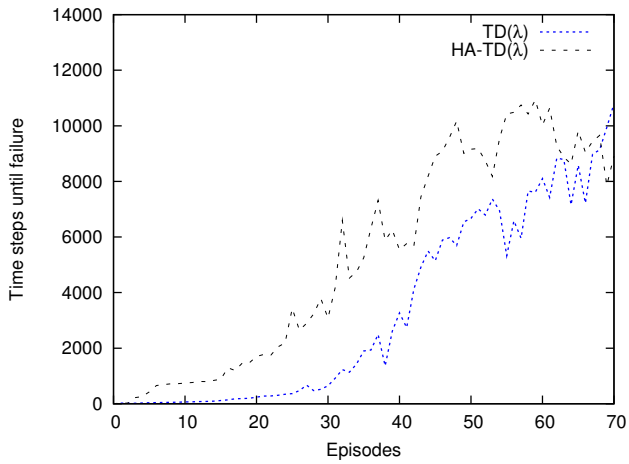

Figure 4. Evolution of the number of steps until failure for the cart-pole problem. This is the average of 100 trials, therefore it is not possible to see individual results that reach the success criterion of 500.000 steps without failure.

probability function:

$$Pr(a) = \frac{1}{1 + e^{V(y_t,\theta_t)+H(y_t,\theta_t)}}, \tag{24}$$

where if the rounding of $Pr(a)$ equals zero, the force applied is positive, if the rounding is equal to one, the force is negative. It can be seen that the heuristics, in this case, is combined with the Value Function $V(y_t, \theta_t)$ inside the rule that is used by the TD(λ) algorithm (which is not the ϵ-greedy rule).

To run our experiments, we used the simulator distributed by Sutton and Barto [12], which implements the function approximator described in Barto *et al.* [2]. Trials consisted of 100 episodes. The goal is to keep the pole without falling for 500.000 steps, in which case the trial terminates successfully. The parameters used were the same as in Barto *et al.* [2]. The value of η used by the HATD(λ) is 10. The reward is -1 upon failure. The pole is reset to vertical after each failure.

Table 2 shows the results obtained (average of 100 trials). One can see that both, the number of the episode in which the pole was successfully controlled and the number of steps needed to learn to balance the pole is smaller in HATD(λ). Figure 4 shows the number of steps until failure for both algorithms. It can be seen that at the beginning, the HATD(λ) presents a better performance, and that both algorithms became similar as they converge to the optimal policy.

Finally, Student's t–test was used to verify the hypothesis that the use of heuristics speeds up the learning process. The results confirm that HATD(λ) is significantly better than TD(λ) until the 50^{th} episode, with a confidence level greater than 95%.

Algorithm	First Successful Episode	Steps until 1st success
TD(λ)	67 ± 16	$1{,}115{,}602 \pm 942{,}752$
HATD(λ)	23 ± 14	$637{,}708 \pm 237{,}398$

Table 2. Results for the cart-pole problem.

7 Conclusion

In this work we presented new theoretical results for the class that encompasses all heuristics-based algorithms, called Heuristically Accelerated Reinforcement Learning. We also have contributed three new learning algorithm, HA-Q(λ), HA-SARSA(λ) and HATD(λ), the first algorithms that uses both heuristics and eligibility traces. Empirical evaluation of these algorithms in the mountain-car and cart-pole problems were carried out.

Experimental results showed that the performance of the learning algorithm can be improved even using very simple heuristic functions. An important topic to be investigated in future works is the use of generalization in the value function space to generate the heuristic function.

ACKNOWLEDGEMENTS

Reinaldo Bianchi acknowledges the support of the FAPESP (grant number 2012/04089-3). Carlos Ribeiro is grateful to FAPESP (2012/10528-0 and 2011/17610-0) and CNPq (305772/2010-4) and Anna Costa is grateful to FAPESP (2011/19280-8) and CNPq (311058/2011-6).

References

[1] J. S. Albus, 'A new approach to manipulator control: The cerebellar model articulation controller (CMAC)', *Trans. of the ASME, J. Dynamic Systems, Measurement, and Control*, **97**(3), 220–227, (1975).

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson, 'Neuronlike elements that can solve difficult learning control problems', *IEEE Transactions on Systems, Man, and Cybernetics*, (13), 834–846, (1983).

[3] Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro, and Anna H. R. Costa, 'Heuristically Accelerated Q-learning: a new approach to speed up reinforcement learning', *Lecture Notes in Artificial Intelligence*, **3171**, 245–254, (2004).

[4] 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).

[5] Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro, and Anna Helena Reali Costa, 'Heuristic selection of actions in multiagent reinforcement learning', in *IJCAI*, ed., Manuela M. Veloso, pp. 690–695, (2007).

[6] Reinaldo A. C. Bianchi, Raquel Ros, and Ramon López de Mántaras, 'Improving reinforcement learning by using case based heuristics', in *Lecture Notes in Computer Science*, **5650**, pp. 75–89. Springer, (2009).

[7] A. Burkov and B. Chaib-draa, 'Adaptive play Q-learning with initial heuristic approximation', in *ICRA*, pp. 1749–1754. IEEE, (2007).

[8] M. L. Littman and C. Szepesvári, 'A generalized reinforcement learning model: convergence and applications', in *ICML'96*, pp. 310–318, (1996).

[9] Andrew Moore, 'Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces', in *Proceedings of the Eighth International Conference on Machine Learning*, (June 1991). Morgan Kaufmann.

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

[11] S. P. Singh, *Learning to solve Markovian Decision Processes*, Ph.D. Dissertation, University of Massachusetts, Amherst, 1994.

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

[13] R. S. Sutton, 'Learning to predict by the methods of temporal differences', *Machine Learning*, **3**(1), 9–44, (1988).

[14] R. S. Sutton, 'Generalization in reinforcement learning: Successful examples using sparse coarse coding', in *Advances in Neural Information Processing Systems*, **8**, pp. 1038–1044. The MIT Press, (1996).

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