USE OF HEURISTICS FROM DEMONSTRATIONS TO SPEED UP REINFORCEMENT LEARNING

Danilo H. Perico*, Reinaldo A. C. Bianchi*

*Centro Universitário da FEI
Av. Humberto de Alencar Castelo Branco, 3972
São Bernardo do Campo, São Paulo, Brasil

Emails: danilo.perico@gmail.com, rbianchi@fei.edu.br

Abstract— The Reinforcement Learning is a well known method for solving problems where the agent needs to learn through direct interaction with the environment. However, this technique is not efficient enough, due to its high computational cost. This work proposes and test the Reinforcement Learning accelerated by heuristics obtained through demonstrations algorithm, in the domain of robotic navigation. Experiments were made in simulated environment using software Player/Stage. Experimental results allowed to conclude that the proposed algorithm can improve the learning performance, reducing significantly the number of steps needed to reach the goal, when compared to the conventional Reinforcement Learning algorithms.

Keywords— Artificial Intelligence, Machine Learning, Reinforcement Learning, Learning from Demonstration, Robotics.

1 Introduction

Robot's learning ability has been pursued by Artificial Intelligence (AI) researchers for several years. Reinforcement Learning (RL) can be quoted as one of the most usual type of learning for this purpose.

RL is a Machine Learning method, in which the agent can learn through interactions with its environment, without any previous knowledge (Sutton, 1988). Furthermore, Reinforcement Learning allows the agent to learn in a unknown environment via exploration. Depending on its performance, the agent receives a reward or penalty, that make the agent learns a set of actions to the optimal behavior. This set of actions is called optimal policy (π^*).

The main issue concerning to the RL refers to the large number of iterations needed by the algorithm to converge, it means that the complete learning process is very slow. So, methods are needed to speed up the learning, even more when this technique is used in the field of autonomous mobile robots.

This work presents an efficient learning method for autonomous mobile robots, to be used in large and complex environments. Being inspired by the learning split into two phases, proposed by Smart and Kaelbling (2002), this paper purposes the combination between the Heuristically Accelerated Reinforcement Learning (HARL) (Bianchi et al., 2008) and the Learning from Demonstration (LfD)(Schaal, 1997), in order to develop the Heuristics from Demonstrations to Accelerate Reinforcement Learning (HfDARL) in the domain of the autonomous mobile robotics.

This paper is organized as follows: sections 2 and 3 briefly reviews RL and some approaches to speed up RL. Section 4 presents a review of the LfD. Section 5 presents the proposal of this work. The results obtained in the simulated environemnt using software Player/Stage are shown in section 6. Finally, Section 7 provides the conclusions and indicates avenunes by which the research proposed in this paper can be extended.

2 Reinforcement Learning

RL algorithm works by using a scalar reinforcement signal $r_{(s,a)}$ (reward or penalty) that is received by the agent in each state s_t update. Each action a_t taken by the agent is defined in order to maximize the total reward it receives, using the knowledge obtained until the present moment t.

One way to formalize the RL is the Markov Decision Process (MDP), where the learning can be modeled by a 4-tuple $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}>$, $\mathcal{S}$: is a finite set of states; $\mathcal{A}$: is a finite set of possible actions; $\mathcal{R} : \mathcal{S} \times\ \mathcal{A} \rightarrow \Re$ is a reinforcement function; $\mathcal{T} : \mathcal{S} \times\ \mathcal{A} \rightarrow \Pi(S)$ is a state transition function, being $\Pi(S)$ the state transiction function mapping probability.

The goal of the agent in RL is to learn an optimal policy of actions which maximizes a function, therefore the MDP enables the mathematical modeling of a RL agent.

Watkins (1989) proposed the algorithm Q-Learning, that is considered one of the most popular RL algorithm. The goal of the Q-Learning is to maximize the function Q for each state, in which Q represents the expected return for taking action a when visiting state s and following policy π. The Q-Learning rule is:

$$\begin{aligned} Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r(s_t, a_t) + \\ \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \end{aligned} \quad (1)$$

Where s_t is the current state; a_t is the action

Algorithm 1 Q-Learning

```
Initialize $Q(s,a)$ arbitrarily
  Repeat
    Visit state $s$
    Select an action $a$ according to the action choice rule.
    Execute the action $a$
    Receive the reward $r(s,a)$ and observe the next state $s'$
    Update the values of $Q(s,a)$ according to the Q-Learning rule:
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    Update the state $s \leftarrow s'$
  Until $s$ is terminal

  Where $s'$ is the next state and $a'$ is the action performed in $s'$
```

performed in s_t; $r(s_t, a_t)$ is the reward received after taken a_t at s_t; s_{t+1} is the new state; a_{t+1} is the action performed in s_{t+1}; γ is the discount factor ($0 \leq \gamma < 1$); and α is the learning rate ($0 < \alpha < 1$).

A common strategy adopted to chose the actions during the exploration process is known as *ε–Greedy*. This strategy pursues the action that results in the higher value to Q with probability $1 - \varepsilon$.

The *ε–Greedy* action choice rule, can be written as:

$$\pi(s_t) = \begin{cases} a_{random} & \text{if q} \leqslant \varepsilon, \\ \arg\max_{a_t} \; Q_t(s_t, a_t) & \text{otherwise.} \end{cases} \tag{2}$$

Where q and ε are parameters that define the exploration/exploitation tradeoff; and a_{random} is an action randomly chosen among those available in state s_t.

The complete Q-Learning is shown in algorithm 1.

3 Accelerating Reinforcement Learning

Among a large number of methods for speed up RL, two are important to the development of this work. The first one is the spatial generalization and the second one is the use of heuristics.

Using the spatial spreading concepts, Ribeiro and Szepesvari (1996) proposed the QS algorithm which is a Q-Learning algorithm combined with spatial spreading of action values. The goal of the QS is to spread the function $Q(s,a)$ to others state-action pairs (x,u) not involved into the current iteration, respecting some similarity criteria. The spreading is defined by some prior domain knowledge.

In order to simplify the domain knowledge inserction, Pegoraro (2001) proposed the QSx algorithm that is, basically, the QS algorithm with knowledge inserction by using similarity degrees.

The spreading function that defines the similarity among the state-action pairs is represented by the equation 3.

$$\sigma(s,a,x,u) = \tau^n \tag{3}$$

Where n is the diversity degree among the state-action pairs; and τ is the similarity factor ($0 \leqslant \tau \leqslant 1$).

The similarity among the state-action pairs can be defined by several ways, like by connectivity, by neighborhood and by symmetry. Sometimes, it is also necessary to restrict the similarity to a region of the domain.

The similarity by connectivity can be defined through the number of actions needed to take the agent from state s to x. Thereby, this similarity can be molded as $f_x(x,s) = \tau^{n_a}$, where n_a is the minimum number of actions needed to go from s to x. The similarity by neighborhood considers that positions topologically close from each other can be defined as similar. The similarity by neighborhood function can be written as $f_x(x,s) = \tau^{d_{s,x}}$, where $d_{s,x}$ represents the distance between states s and x. Finally, in the similarity by symmetry, an experience performed in the state s can be reflected to the state x, since these states are simmetryc (Pegoraro, 2001).

Another important way to speed up the RL, relevant to this work, is the use of heuristics. A Heuristically Accelerated Reinforcement Learning (HARL) algorithm is a way to solve a MDP problem with explicit use of a heuristic function for influencing the choice of actions by the learning agent (Bianchi et al., 2008).

The heuristic function $H(s,a)$ defines 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.

An algorithm proposed by Bianchi et al. (2008) to implement the HARL is the Heuristically Accelerated Q-Learning (HAQL). This algorithm is an extension of the Q-learning algorithm with the heuristic function inserction in the *ε-Greedy* action choice rule, that can be written as:

$$\pi(s_t) = \begin{cases} a_{random} & \text{if q} \leqslant \varepsilon, \\ \arg\max\limits_{a_t} \, [Q_t(s_t,a_t) + & \\ \qquad\qquad \xi H_t(s_t,a_t)] & \text{otherwise.} \end{cases} \tag{4}$$

Using heuristics to speed up RL algorithms has been shown to be effective. Several works have used heuristics to speed up learning such as in the case-based study (Celiberto et al., 2011), for example.

4 Learning from Demonstration

Learning from Demonstration (LfD) (Schaal, 1997) is a technique that develops policies from examples provided by a teacher. Examples are defined as sequencies of state-action pairs that are recorded during the teacher's demonstration of the desired behavior.

According to Argall et al. (2009), LfD has been applied to a large variety of problems in the mobile robotics domain, since this learning does not need expert knowledge of the domain dynamics and, furthermore, demonstration is an intuitive way for communication from humans.

LfD can be segmented into two fundamental phases: gathering the examples and deriving a policy from the examples. The first phase comprises in the execution of the demonstrations by a teacher. The second phase comprises of deriving a policy that reproduces the demonstrated behavior, utilizing the dataset of examples obtained in the first phase.

LfD systems are completely limited by the quality of the dataset obtained in the demonstration (first phase). Argall et al. (2009) quoted two main reasons that are responsible for poor learner performance: dataset sparsity, and poor quality of the dataset examples, outcome from a teacher's inability to perform the task.

One existing method that can be used to reduce the LfD limitations, proposed by Smart and Kaelbling (2002), allows the combination of the LfD and the RL. This method works in two phases: demonstration and exploration. In the demonstration the $Q(s, a)$ function is updated while the examples are demonstrated to the agent. After that, the agent uses the exploration via RL using Q with the previously values obtained in the first phase.

Nevertheless, as the Q function is a iterative function, generally a large number of demonstrations are needed to improve substantially the agent performance.

5 Heuristics from Demonstrations to Speed Up Reinforcement Learning

As quoted before the RL has been presented as one of the most popular manner to implement robot's learning ability. However, RL is a slow method of learning and may not be efficient when the domain contains dynamic agents and large environments, even taking some of the previous techniques to speed up the learning. A similar issue may be pointed in the LfD, once the derived policy can not change anymore after the demonstration phase. This may be a problem in a environment that can suffer layout changes. Also, depending on the quality of the demonstrations, the policy derived in the LfD may not be the optimal one.

So, the goal of this work is to propose and test an algorithm that can keep the main advantages of the RL, like the exploration, and the main advantages of the LfD, like the fact of being an intuitive communication way between humans and robots.

Inspired by the article presented by Smart and Kaelbling (2002), this work proposes the combination of the Heuristically Accelerated Reinforcement Learning (Bianchi et al., 2008) and the Learning from Demonstration (Schaal, 1997). This fusion can be achieved by using the demonstrations as a method to obtain the heuristics instead of the Q function. So, the heuristics can accelerate the RL and the algorithm keeps the RL features and the advantages of the LfD. This fusion of techniques is named here "Heuristics from Demonstrations to Accelerate Reinforcement Learning" (HfDARL).

In this model, the heuristics are created while the demonstrations are being made, through the update of the H function in each visited state-action pair. So, for each action taken by the teacher, a maximum value η is granted to $H(s, a)$, where $\eta \in \Re$ e $0 < \eta \leqslant \infty$.

As the purpose of this work is to improve the agent performance with the minimum number of demonstrations, even considering a large and complex environment, it is necessary to use a generalization technique to achieve the best heuristic with few paths examples.

One approach to deal with the generalization is by using the QSx algorithm (Pegoraro, 2001), as already said. As the QSx can be an efficient technique to execute the spatial spreading of the Q function, the idea here is to adopt its concepts in order to spread the H function instead of the Q one.

The similarity applied to the heuristics will be measured by the spreading function 3. Thereby, the H function spreading rule wil be:

$$H(x, u) = H(s, a)\sigma_H(s, a, x, u) \tag{5}$$

Where $H(s, a)$ represents the maximum value η applied as heuristic in the state s to the action a.

Once the heuristic is obtained through the data collected during the demonstrations, the learning must occur using the Heuristically Accelerated Reinforcement Learning. The complete proposed algorithm is named here Heuristics from Demonstration to Accelerate Q-Learning (HfDAQL) and is shown in algorithm 2.

6 Experimental Results

In order to verify the behavior of the proposed learning, experiments were performed considering the following algorithms: HfDAQL (proposed in

Algorithm 2 HfDAQL

```
Initialize $H(s,a) = 0$.
  Repeat
    Repeat
      Demonstrate to the learner agent which action $a$ take in state $s$
      Update $H(s,a)$ with the maximum value $\eta$
      if $H(x,u) < H(x'',u'')$ then
        Spread $H(s,a)$ to the similar state-action pairs, considering $H(x,u)$ according to the spreading rule:
        $H(x,u) \leftarrow H(s,a)\sigma_H(s,a,x,u)$
      end if
    Until the target is reached
  Until The complete number of demonstrattions is reached
Initialize $Q(s,a)$ arbitrarily
  Repeat
    Visit state $s$
    Select an action $a$ according to the HAQL action choice rule.
    Execute the action $a$
    Receive the reward $r(s,a)$ and observe the next state $s'$
    Update the values of $Q(s,a)$ according to the Q-Learning rule:
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    Update the state $s \leftarrow s'$
  Until $s$ is terminal

  Where $s'$ is the next state and $a'$ is the action performed in $s'$, $(x,u)$ represents the state-action pairs which are not directly involved in the current demonstration and $H(x'',u'')$ represents the heuristics values received in previous demonstrations to $(x,u)$.
```

this work), HAQL (with heuristics previously defined *ad hoc*) (Bianchi et al., 2008) and the Q-Learning (Watkins, 1989). Thereby, the proposed algorithm performance could be compared with others two well known learning algorithms.

The experiments were performed in the autonomous mobile robotics domain by using a simulated environment implemented in software Player/Stage.

Player/Stage is a robot simulation tool: Player is a Hardware Abstraction Layer and Stage is a plugin to Player which listens to what Player is telling it to do and turns these instructions into a simulation (Owen, 2010).

To perform the simulations, the robot Pioneer 2-DX, made by Adept MobileRobots, was chosen. This is a mobile robot capable to move itself through flat grounds, preferentially in closed environments. The robot has 16 sonars that will be used to avoid collisions. In this work the robot localization will be made by an ideal odometry, that ignores odometry errors. This kind of approach is possible because Player/Stage has a feature that allows that.

The virtual environment was established based on a real laboratory, located at Centro Universitário da FEI. This room has dimensions 10 x 7.25 m, is enclosed by walls and has desks and lockers inside. The environment and the robot with its sonars range can be seen in figure 1.

In order to perform the simulation, the environment was discretized into square cells, each cell with 25 x 25 cm of dimension. So, a grid with 40 x 29 cells was created. The robot orientation was partitioned into 16 discrete values, multiples of the 22.5^o. Thus the variables x, y and θ, used for learning, could describe a rough approximation of the robot real state.

Three actions were defined as executable by the robot: move ahead and turn either clockwise or anticlockwise. The actions are executed until the state s change. Additionally to these actions, the robot can move backward, but it is a passive action automatically executed when the robot is very close to an obstacle (less than 25 cm from a lateral obstacle and less than 30 cm from a frontal obstacle). This passive action results in negative reinforcement to the agent.

The parameters used in all methods applied in this section are the following: reinforcement of -50 when the robot needs to move backward, -1 for any executed action, instead of move backward or reach the goal and 1000 for reaching the goal state; learning rate α: 20%; discount factor γ: 90%; and exploration rate ε: 20%.

The robot starts each training episode from a random pose, and the goal corresponds to a region in the lower right corner (defined as the region x > 10,00 m; 0,50 m > y > 0,75 m).

The heuristics were defined as follows: for the HAQL with heuristics *ad hoc*, it was granted to $H(s,a)$ value equal to 5 to the action move forward when the robot is going to east or southeast (using global coordinates in order to simplify the understanding). All other H non-related to these

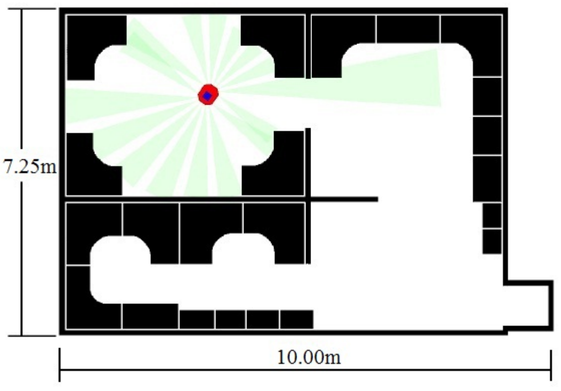


Figure 1: Virtual laboratory and robot.

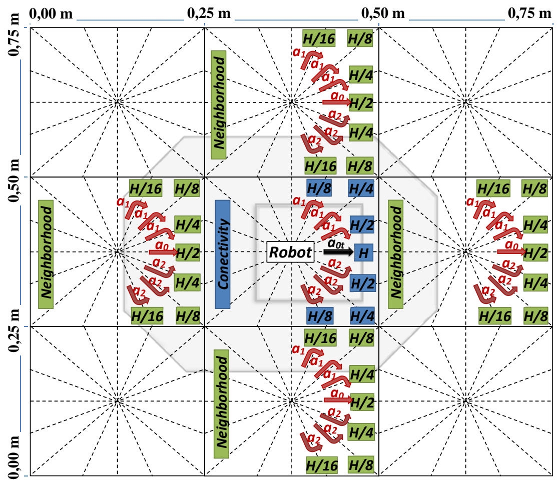


Figure 2: Heuristics spatial spreading rule.

coordinates received 0. For the HfDAQL algorithm the demonstrations are responsible for creating the heuristics, so the maximum value η was granted to $H(s, a)$ to the state-action pairs used during the demonstrations. Besides that, H values were spread to others state-action pairs that were not involved directly in the demonstrations, using spatial spreading concepts with similarity by connectivity and neighborhood.

Thereby, η was defined as 10, following the precepts established by Bianchi et al. (2008).

Spatial spreading was performed as shown in figure 2, where the grid is represented by continous lines and the orientation discretization by the dashed lines. The spreading was performed with the use of similarity by conectivity in the robot's current state, considering action a_{0t}, and with the use of similarity by neighborhood in the neighbor cells. It was used the spread function (equation 3) with τ equal to 0.5 and n equal to the number of actions needed to take the agent from s to x (conectivity) or the distance between s and x (neighborhood).

In this experiment, demonstrations consist of guiding the robot from a random initial position to the goal by teleoperation and the teacher has a global vision of the system. Demonstrations are considered suboptimals when the teacher tries to drive the robot directly to the proposed goal. These demonstrations are not considered optimals because, as the teacher is the responsible for the path choice, this path may not be the best one. However, the difference between optimal and suboptimal is quite subtle.

Thus, a study was made in order to verify how the quantity and quality of the demonstrations influence the learning in the proposed algorithm (HfDAQL). The study showed that the influence of the demonstrations is more related to the quality and less to the quantity. Quality here refers to the covered area during the demonstrations, taking as basis that demonstrations always aim to drive the robot directly to the goal state.

Once the demonstrations influence study was done, it is possible to go ahead with the HfDAQL complete analysis. So, in order to proceed with the experiments, some parameters were defined: 5 suboptimal examples of paths were used to cover 64% of the free area; 500 episodes were performed and 10 training sessions. The results can be seen in figure 3. It is possible to see in the graph that HfDAQL needs less steps to learn. This difference is more striking in the first episodes, where the proposed algorithm is much more efficient.

It was also made a Student T-test for the data presented in Figure 3 that can be seen in Figure 4. Notice that in the first episodes the majority of the results are above the dashed line (5% upper bound), which means that the algorithms compared to the HfDAQL are significantly different, with a confidence level of at least 95%.

Figure 5 shows the paths of the robot with the 3 algorithms used in this work. The three figures are created as result of the 1^o training episode. Besides the visual comparison, it is relevant to highlight the time needed by each algorithm to complete this episode: Q-Learning took 3.5 hours, HAQL *ad hoc* took 33 minutes and the proposed HfDAQL less than 5 minutes.

7 Conclusion

This work has proposed the use of demonstrations to obtain the heuristics used to accelerate the RL. The results allow us to conclude that the proposed algorithm, HfDAQL, was more efficient, mainly in the first training episodes, when compared to the Q-Learning and the HAQL with heuristics defined *a priori ad hoc*. It happens because the proposed algorithm limits the search space explored by the learner agent.

Some advantages of the HfDAQL are: the

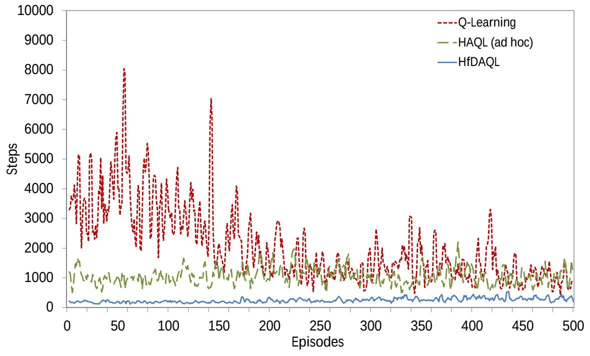


Figure 3: HfDAQL in comparison to Q-Learning and HAQL with heuristics ad hoc.

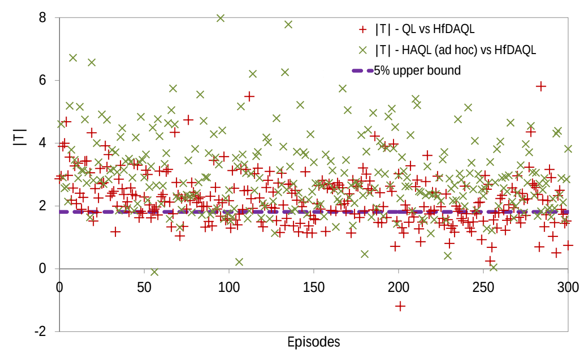


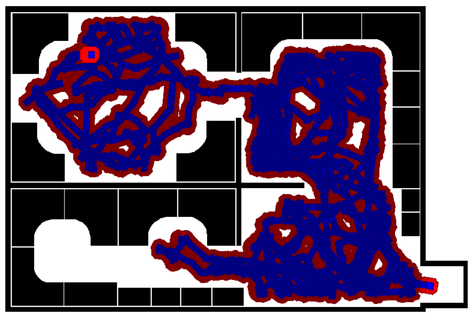
Figure 4: Results of the Student T-test for the algorithms analyzed.

possibility to eliminate detailed knowledge of the robot's sensors during programming, the low number of demonstrations needed to speed up substantially the learning and the reduced number of steps needed to reach the goal since the first learning episodes.

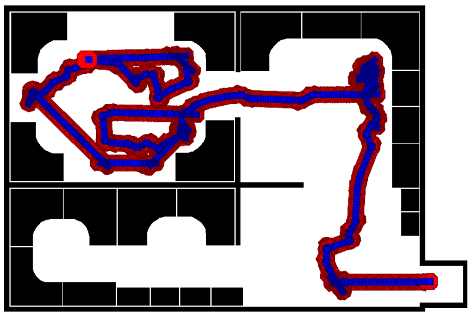
The main contribution of this work is the proposal of the HfDAQL algorithm, that uses heuristics obtained from demonstrations to speed up the RL. Future works include studying the HfDAQL in more complex environments by implementing the algorithm in a real robot and studying others spreading functions to the H function.

8 Acknowledgements

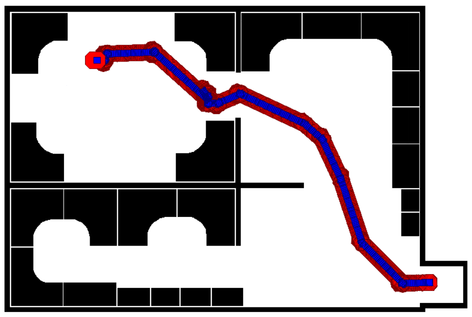
Reinaldo A. C. Bianchi acknowledges the support of the FAPESP (grants 2011/19280-8, 2012/04089-3 and 2013/17286-4).



(a) Q-Learning



(b) HAQL (ad hoc)



(c) HfDAQL

Figure 5: Robot footprint in the 1^o episode.

References

Argall, B. D., Chernova, S., Veloso, M. and Browning, B. (2009). A survey of robot learning from demonstration, *Robot. Auton. Syst.*.

Bianchi, R. A. C., Ribeiro, C. H. C. and Costa, A. H. R. (2008). Accelerating autonomous learning by using heuristic selection of actions, *Journal of Heuristics*, pp. 135–168.

Celiberto, L. A., Matsuura, J. P., de Mántaras, R. L. and Bianchi, R. A. C. (2011). Using cases as heuristics in reinforcement learning: A transfer learning application, *IJCAI*.

Owen, J. (2010). *How to use Player/Stage*, York.

Pegoraro, R. (2001). *Agilizando aprendizagem por reforço em robótica móvel através do uso de conhecimento sobre o domínio*, PhD Thesis, Universidade de São Paulo.

Ribeiro, C. H. C. and Szepesvari, C. (1996). Q-learning combined with spreading: Convergence and results, *Proceedings of ISRF-IEE International Conference on Intelligent and Cognitive Systems*, Tehran, Iran.

Schaal, S. (1997). Learning from demonstration, *Proceedings of Advances in Neural Information Processing Systems*, MIT, MA, USA.

Smart, W. D. and Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation*, Washington, D.C., USA, pp. 3404–3410.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning*.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*, PhD thesis, King's College, Cambridge, UK.