Homeokinetic Reinforcement Learning

Simón C. Smith and J. Michael Herrmann

Institute of Perception, Action and Behaviour, School of Informatics,
The University of Edinburgh, 10 Crichton St, Edinburgh, EH8 9AB, U.K.
{artificialsimon,michael.herrmann}@ed.ac.uk

Abstract. In order to find a control policy for an autonomous robot by reinforcement learning, the utility of a behaviour can be revealed locally through a modulation of the motor command by probing actions. For robots with many degrees of freedom, this type of exploration becomes inefficient such that it is an interesting option to use an auxiliary controller for the selection of promising probing actions. We suggest here to optimise the exploratory modulation by a self-organising controller. The approach is illustrated by two control tasks, namely swing-up of a pendulum and walking in a simulated hexapod. The results imply that the homeokinetic approach is beneficial for high complexity problems.

1 Introduction

Reinforcement Learning, discrete [1,12,13] as well as continuous [5], aims at solving dynamical optimisation problems. For this purpose a utility function and/or a control policy is constructed. Optimal performance can be reached asymptotically under certain conditions. However, because often Markovian state transitions and slow decay of the learning rate cannot be asserted in practical problems, only suboptimal solutions are found.

Additionally, in high dimensions, the exploration of the state space is time consuming. Gradient-based reinforcement learning can speed-up the optimisation process, but is prone to local optima, and if the gradient is not known then probing actions must be used in order to obtain gradient information. High-frequency probing [14] tests two alternative actions virtually at the same time which seems appropriate for an autonomous agent which may not be able to apply different actions in the same state. In addition, the set-up of the probing actions requires some domain knowledge and becomes cumbersome in high dimensions. Furthermore, what priorities should be used when sequentially probing the manifold of behaviours in robots with many degrees of freedom? A robot with a reinforcement learning (RL) controller is biased to keep trying the path that is expected to give him the best reward in the future, thus seemingly non rewarding nearby states are less likely to be explored.

We propose to use an auxiliary algorithm that learns to probe the system. For this purpose we will not follow the gradient of the utility function, but will aim at maximising the learning success achieved by the probing actions. This will help to obtain a more reliable representation of the utility function in shorter time, while the reinforcement learning component will be responsible for the actual increase of the expected reward.

The probing algorithm relies on a self-organising (SO) control paradigm described in Ref. [10]. The SO controller generates motors signals based on estimated next sensor values. This signal will be used by the reinforcement learning controller as exploration mode and to update the parameters values in an actor-critic configuration. A representation of the world and the controller are modelled and updated based on the time loop error, see [3,4]. In Fig. 1 a scheme of the architecture can be seen, the reinforcement learning controller generates a motor signal u_t given the actual states x_t and the exploratory signal n_t provided by the SO controller. Given the actual motor signal and the actual state the model predicts the next sensor input which is used to calculate the time loop error and to update the SO controller. It is essential to the approach followed here that the full loop through the environment is monitored by the robot. This loop can be represented by a map of previous to new sensor values, but as well also as a map from previous to new motor commands. The latter case is actually more convenient if as often the dimensionality of the motor space is lower than that of the sensor space.

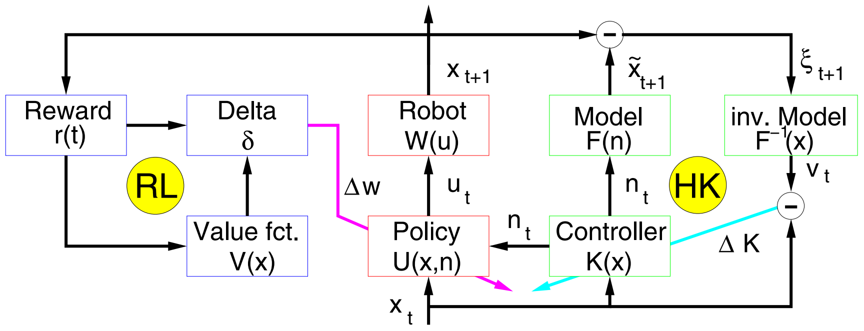

Fig. 1. Architecture of the sensorimotor loop (RL: Reinforcement learning controller, HK: Homeokinetic controller; for other symbols see Section 2)

We present a comparison of our approach with a standard version of continuous reinforcement learning [5] in the low dimensional task of swinging up a pendulum with limited torque and in an hexapod robot with twelve degrees of freedom where walking speed is to be optimised.

2 Reinforcement Learning in Continuous Space and Time

Following [5], the control command is given by

$$u_t = U_t\left(x_t\right) = s\left(A\left(x_t; w^A\right) + \sigma \mathbf{n}_t\right), \tag{1}$$

where s is the output function, $\mathbf{n}$ is a probing input of strength σ and

$$A\left(x_t; w^A\right) = N\left(x\right) \sum_i w_i^A \exp\left(-\frac{\left\|x_t - \mu_i\right\|^2}{2\rho_i^2}\right) \tag{2}$$

is a policy function that depends on parameters w^A (ρ_i and μ_i are assumed to be fixed). The factor $N\left(x\right) = \left(\sum_i \exp\left(-\frac{\|x_t-\mu_i\|^2}{2\rho_i^2}\right)\right)^{-1}$ normalises the actor output. The parameters w^A are updated according to

$$\Delta w_i^A = \eta^A \delta_t \mathbf{n}_t \frac{\partial A\left(x_t; w^A\right)}{\partial w_i^A}, \tag{3}$$

where η^A is a learning rate.

While the last term in (3) is easily obtained from (2), the essential part of this learning rule includes the correlation of the probing input $\mathbf{n}$ and the delta error

$$\delta_t = r_t - \frac{1}{\tau} V_t + \dot{V}_t \tag{4}$$

The utility function V is represented by another parametrised function which is simultaneously updated.

There are various ways of choosing the probing excitation of the robot control in Eq. 1. Gullapalli [7] suggested to use noise while others [14,2] have proposed high-frequency oscillatory modulations of the motor command. Our experiments confirm that the type of the probe does not matter in low-dimensional problems. For robots with many degrees of freedom, the dynamics of the correlation among the degrees of freedom of the controlled system becomes crucial such that the choice of the probing stimulus becomes non-trivial. In high-dimensional problems it is obviously not possible to test all actions in all states infinitely often as it would be required in discrete reinforcement learning algorithms. Also for continuous algorithms orienting the exploration to promising directions is essential. We propose to use an approach in the present context that we have previously developed in a different setting [8].

3 Learning in Motor Space

Instead of using noisy probing, we propose to modulate the motor command (1)

$$u_t = s\left(A\left(x_t; w^A\right) + \sigma K\left(x_t\right)\right) \tag{5}$$

by an exploratory controller

$$K\left(x_t\right) = g\left(C x_t + C_0\right). \tag{6}$$

This controller receives the current sensory input vector x_t and determines the direction of exploration in dependence on the multidimensional parameters $C \in \mathbb{R}^{m \times n}$ and $C_0 \in \mathbb{R}^m$ and a further nonlinear function g. In order to adapt the parameters C and C_0, the new sensory inputs are compared with a prediction $\hat{x}$ by a world model M based on previous inputs or outputs. For simplicity we use a linear predictor that uses only the motor commands (5) and receives thus information about previous inputs only indirectly.

$$\hat{x}_{t+1} = M\left(u_t\right) = Du_t + D_0 \tag{7}$$

The comparison of the corresponding sensory input x_{t+1} and its estimate by the internal model $\hat{x}_{t+1}$ results in the prediction error $\xi_{t+1} = \hat{x}_{t+1} - x_{t+1}$ which is a vector in the perceptual space.

In order to formulate a learning rule for the exploratory controller (7) we will follow the procedure in Ref. [8] and express the error in the motor space which can be achieved by defining a transformed error η_t via

$$M\left(u_t\right) + \xi_{t+1} = M\left(u_t + \eta_t\right). \tag{8}$$

Because $M\left(u_t\right) + \xi_{t+1} = x_{t+1}$, the motor error η can be interpreted as the control correction required to compensate the inaccuracy of the model M. η is a retrospective error that can be determined only after the event of receiving the new stimulus x_{t+1}. Nevertheless, minimisation of η is a relevant goal for the adaptation of the system. The definition (8) is implicit and may be empty which calls for the use of a regularised inverse of M to explicitly obtain an approximation of η. Practically, Eq. 8 is transformed into a motor level error exploiting the assumed linearity of the model (7),

$$\eta_t = M'^{+} \xi_{t+1}, \tag{9}$$

where M'^{+} is the pseudo-inverse of the derivative of the model (7), i.e. the pseudoinverse of D in Eq. 7. In analogy to Ref. [3] this defines a homeokinetic error function in the motor space

$$E_t = \eta_t^\top \left(J_t J_t^\top\right)^{-1} \eta_t \tag{10}$$

where J is the Jacobian of the sensorimotor loop, see below. We are going to perform a gradient descent with respect to this error function in order to adapt the parameters of the controller (6).

To calculate the Jacobian, we use the derivatives $M'_u = D$ and $U'_x = s' \circ \left(\frac{\partial A}{\partial x} + \sigma g' \circ C\right)$ such that we find from $J_t = \phi'_u = U'_x\left(x_t\right) M'_u\left(x_{t-1}; u_{t-1}\right)$[1]

$$J_t = \left(\frac{\partial A}{\partial x} + \sigma g'_t \circ C_t\right) D_t.$$

This gives rise to the following formulation of the shift ν, i.e. the change in motor command that would have been required to correctly predict the following motor command, namely

$$\nu_{t-1} = J_t^{-1} \eta_t$$

[1] The dependence $\frac{dx}{du}$ of the new sensory input on the motor command is approximated here based on the assumption of a correct model (certainty equivalence). The symbol $\circ$ denotes component-wise multiplication.

While the above interpretation (9) of η as retrospective error connects sensor and motor space, we have here a connection between the two points in time within the motor space that reflects the dynamical properties of the full sensorimotor loop. The error function (10) becomes thus simply

$$E_t = \nu_{t-1}^\top \nu_{t-1}$$

which lead to a convenient update rule of the controller matrix C. Omitting the time indices we find

$$\frac{1}{\varepsilon_C} \Delta C = -\frac{\partial E}{\partial C} = -2\nu^\top \frac{\partial \nu}{\partial C} = 2\nu^\top J^{-1} \frac{\partial J}{\partial C} J^{-1} \eta - 2\nu^\top J^{-1} \frac{\partial \eta}{\partial C}$$

using the rule $\frac{\partial Y^{-1}}{\partial X} = -Y^{-1} \frac{\partial Y}{\partial X} Y^{-1}$. The derivative $\frac{\partial \eta}{\partial C}$ cannot be determined, because we have no information of the dependence of the prediction error on the controller parameters, therefore we set $\frac{\partial \eta}{\partial C} = 0$ and are left with

$$\frac{1}{\varepsilon_C} \Delta C = 2\nu^\top J^{-1} \frac{\partial J}{\partial C} J^{-1} \eta = 2\nu^\top J^{-1} \frac{\partial J}{\partial C} \nu$$

where

$$\frac{\partial J_t}{\partial C} = \frac{\partial}{\partial C} \left(\frac{\partial A}{\partial x} + \sigma g_t' \circ C_t \right) D_t.$$

We may ignore the effect of the controller on the sensitivity of the actor in the reinforcement learning component, i.e. set $\frac{\partial}{\partial C} \frac{\partial A}{\partial x} = 0$. We may also assume that the details of the actor are not specified by the reward but will follow essentially the homeokinetic control. In this case the term $\frac{\partial}{\partial C} \frac{\partial A}{\partial x}$ is parallel to the remainder and the resulting numerical factor can be absorbed into the learning rate. We have thus arrived at essentially the same learning rule as in Ref. [8],

$$\frac{1}{\varepsilon_C} \Delta C = \chi \left(D\nu \right)^\top - \chi^\top \frac{\partial g'^{-1} \circ \eta}{\partial C},$$

which, however, is to be evaluated at the controller with the reinforcement learning component.

Inserting the correct time indexes we obtain

$$\frac{1}{\varepsilon_C} \Delta C = \chi_{t-1} \left(D_t \nu_{t-2} \right)^\top - 2 \left(\chi_{t-1} \circ g_{t-2} \circ \left(g_{t-2}' \right)^{-1} \circ \eta_{t-1} \right) x_{t-2}^\top$$

with $\chi_{t-1} = \left(R_t^\top \right)^{-1} \nu_{t-2}$. The update rule for C_0 can be found analogously,

$$\frac{1}{\varepsilon_C} \Delta C_{0,t} = -2 \left(\chi_{t-1} \circ g_{t-2} \circ \left(g_{t-2}' \right)^{-1} \circ \eta_{t-1} \right)$$

4 Homeokinetic Reinforcement Learning: Experiments

In order to test our approach two nonlinear control task were implemented. The first one is the pendulum swing-up task (number of sensors $n = 2$, number of motors $m = 1$) where a pendulum has to be brought to the upright position to obtain the maximum reward. The second task consist in teaching an hexapod robot ($n = 12$, $m = 12$) to walk based on a measure of the overall speed. Both pendulum and robot are realised in the LpzRobots simulator [11].

For comparability, we follow the procedure in Ref. [5] and compare the performance to an RL controller configured as an actor-critic, see Eq. 1 for the actor output, and Eq. 3 for the learning rule. The critic is approximated by the relation $\dot{V}(t) \cong (V(t) - V(t - \triangle t))/\triangle t$ using the backward Euler approximation, which rises from the error signal (4)

$$\delta(t) = r(t) + \frac{1}{\triangle t}\left[\left(1 - \frac{\triangle t}{\tau}\right) V(t) - V(t - \triangle t)\right]. \tag{11}$$

The update of the w_i follows a gradient descent with respect to δ.

$$\dot{w}_i = \eta^C \delta(t) \frac{\partial V(x(t - \triangle t); w)}{\partial w_i}, \tag{12}$$

where η^C is a learning rate.

Actor and critic functions are implemented as a normalised Gaussian network. The sigmoid function is defined as $s(x) = \frac{2}{\pi} \arctan\left(\frac{\pi}{2} x\right)$. In the classic RL approach we use coloured noise with a correlation length of 0.1 as probing input with strength σ, in the case of self-exploring RL controller the same value σ is used to weigh the output $\mathbf{n}$ of the SO controller. The strength of the probing signal is weighted by σ, following the idea of [7], while the reward become bigger the probing input should become weaker, the value is calculated by $\sigma = \sigma_0 \min\left\{1, \max\left\{0, \frac{V_1 - V(t)}{V_1 - V_0}\right\}\right\}$ where V_0 and V_1 are the minimal and maximal levels of the reward. For the SO controller the activation function $g(\cdot) = \tanh(\cdot)$ is used.

4.1 Performance in a Toy Example

In a pendulum swing-up task, we use the same configuration as in Ref. [5] where the actor and the critic function are implemented in a 15×15 grid with the angle $\theta \in [-\pi, \pi]$ against the vertical line and the angular velocity $\omega \in [-2\pi, 2\pi]$. The reward function $r(\theta) = \cos(\theta)$ assumes the maximum at the upright and the minimum at the downward position of the pendulum. Each trial lasts for 20 seconds if $\mid \theta \mid < 5\pi$, otherwise a minimal reward is given for one second and the trial is reinitialised in a random state. The performance of the trial is measured by the time when the pendulum is in the range $\mid \theta \mid < \pi/4$. For the SO controller mostly the same setup is used. This problem is not trivial given the maximum applicable force to the pendulum $u^{\max} < m\gamma l$, this maximum force is multiplied to Eq. 5 with m as the mass of the pendulum, γ as the gravitational constant and l as the length from the pivot to the end of the pendulum, with a small enough $u^{\max}$ the pendulum has to build up momentum to be able to reach the upper position.

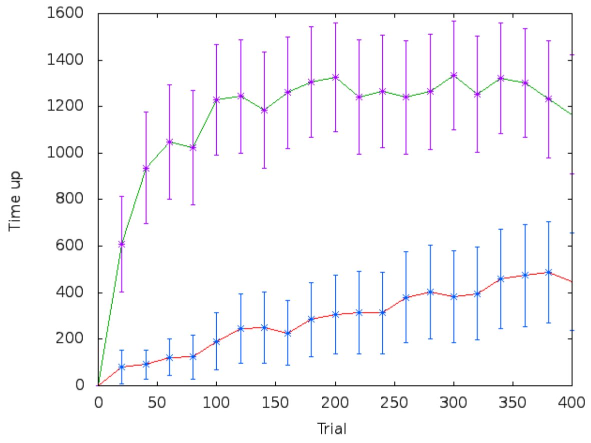

Fig. 2. Average of 50 experiments, each with 400 trials of the swing-up task with RL (upper trace) and with the combined controller (lower trace). The physical parameters are $m = 1$, $\gamma = 9.8$, friction $\mu = 0.01$, $u^{\max} = 5.0$. The parameters for the RL controller are $\tau = 1.0$, $\sigma_0 = 0.5$, $V_0 = -1$, $V_1 = 1$, $\triangle t = 0.02$. The learning rate for the C values were $\epsilon_c = 0.01$. The errorbars indicate deviations over 100 runs for each control task.

Fig. 2 shows the result of the average balance time per trial for 100 experiments with 400 trials each. Results discusion in section 5.

4.2 Self-organisation of Walking in an Hexapod

The hexapod (Fig. 3) resembles a insect with the three pairs of legs, two antennae and a thorax. A two-axis joint is placed where the legs meet the thorax allowing vertical and horizontal rotations, a servo motor actuate over each axis of the joint. In every axis a sensor measures angle θ with respect to the initial position and another sensor measures the angular velocity ω of the leg with pivot on the joint with the thorax. The joint between the femur and the tibia rotate in one axis with a damping action as springs.

The task of the hexapod robot was to improve its overall speed. A reward function is directly proportional to the speed in the (x,y)-plane. Due to the symmetry of the robot no particular movement direction is implied, i.e. in some trials the robot moved in direction of the antennae and in other ones in the opposite direction. The setup of the experiment is similar to the Pendulum, where a trial of 20 seconds is conducted by the controller, after that time the position and the velocities of each leg are set randomly. The performance of each trial is measured by the average speed of the trial. Again, normalised Gaussian networks are used as basis functions for each axis of the thorax-femur joints with 15×15 centers in the range $\left[-\frac{\pi}{8}, \frac{\pi}{8}\right] \times \left[-\frac{5}{4}\pi, \frac{5}{4}\pi\right]$ for vertical movements and $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \times \left[-\frac{5}{4}\pi, \frac{5}{4}\pi\right]$ for horizontal movements of the legs. This high-dimensional task was performed over 4000 trials with SO controller and with noise probing signal, see Fig. 4.

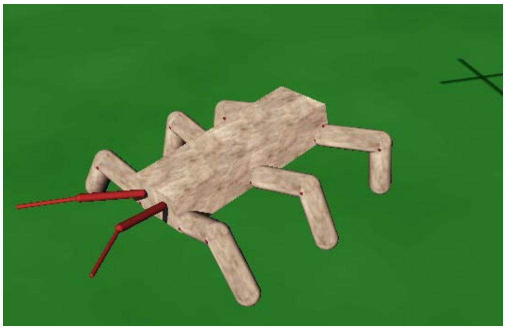

Fig. 3. Hexapod robot realised in the LpzRobots simulator. Joints between legs and thorax have two degrees of freedom and a servo motor for each axis. The joint in the middle of each leg contains only an unactuated spring.

5 Results and Discussion

5.1 Pendulum Results

A comparison of the results for a standard RL controller and for the SO controller are shown in Fig. 2. The swing-up task appears to be easily learned by the RL controller, its slope is steepest, a stable performance is reached earlier, and the total time spent in the upright position is longer. Interestingly, the SO controller never reaches a higher count of maximally rewarded states. This and the evidence that the behaviour is learned, shows that this controller continues to explore new states even if the maximum of the reward function has been already discovered. Because learning is driven merely by the correlation between exploratory action and utility function consistency, the results for this low-dimensional problem are little impressive, whereas in high dimensional tasks, where exploration is a less trivial problem, this SO controller will allow the robot to keep exploring such that local maxima of the expected reward or regions and directions with low gradients can be avoided more easily.

5.2 Hexapod Results

The results for the hexapod with the RL and the SO controller are shown in Fig. 4. As expected, the SO controller shows an advantage in the sense that throughout the experiment the increase of the average speed is higher than the maximal speed that was achieved by the RL controlled robot. It is, moreover, obvious that a stable performance is not clearly achieved by either approach, even after 4000 trials. The relatively high speed is produced by the SO controller even in spite the exploratory tendency of the SO controller, which can be considered as a further advantage of the explorative strategy.

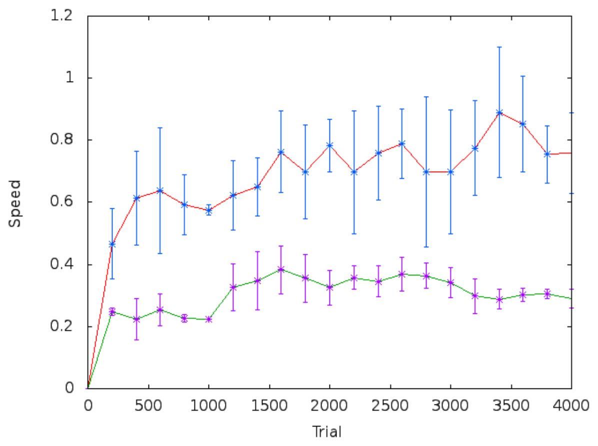

Fig. 4. Average speed for the hexapod with RL controller (lower trace) and with the combined controller (upper trace) during 4000 trials at a learning rate $\epsilon_C = 0.1$. The errorbars indicate deviations over three runs for each control task.

6 Conclusion

We have presented an integration of two approaches to the unsupervised generation of behaviour in robots. The interaction is based on an objective function that maximises the sensitivity of the learning systems with respect to mismatches in the utility function while simultaneously a RL component aims to maximise the future reward. We have tested our approach with two exemplary tasks of different complexity and have shown that

– the exploration induced by the SO controller may counteract the reward maximisation in an optimally tuned low dimensional task, while
– the SO controller seems to aid the learning process by guiding the exploration in a high dimensional task, and that
– the variable coherency of the action modulation in the SO controller improves the capability of the algorithm to escape local minima and flat regions of the goal function.

For comparability of the two variants of learning we ran the experiments with restarts after each trial. This is necessary in RL with random exploration but it is not required in the self-organized variant. If a stable performance is reached at any local or global optimum the sensitivity of the SO controller increases until the state of the systems escapes from the stationary behaviour. Restarting may not be an option for an autonomous robot such that an SO controller may be required here also for practical reasons.

Current work includes the comparison of the quality and frequency of the visited states as well as more systematic assessment of the scaling properties which are promising as we have shown recently in the context of guided self-organisation [9].

Acknowledgement. The design of the hexapod robot has been developed by Guillaume De Chambrier based on the data in Ref. [6]. We are very grateful to Georg Martius for helpful explanations of details of the LpzRobots simulator and to Frank Hesse for comments on the study. We want to thank Friedhelm Schwenker for the perfect organisation of the PSL 2011 workshop where the present paper has been first presented. This research has been funded by The Advanced Human Capital Program of the National Commission for Scientific and Technological Research (CONICYT) of the Republic of Chile.

References

1. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man and Cybernetics 13, 834–846 (1983)
2. Der, R.: Self-organized acquisition of situated behavior. Theory Biosci. 120, 179–187 (2001)
3. Der, R., Michael Herrmann, J., Liebscher, R.: Homeokinetic approach to autonomous learning in mobile robots. VDI-Berichte, vol. 1679, pp. 301–306 (2002)
4. Der, R., Liebscher, R.: True autonomy from self-organized adaptivity. In: Workshop Biologically Inspired Robotics, Bristol (2002)
5. Doya, K.: Reinforcement learning in continuous time and space. Neural Computation 12, 219–245 (2000)
6. Ekeberg, Ö., Blümel, M., Büschges, A.: Dynamic simulation of insect walking. Arthropod Structure & Development 33, 287–300 (2004)
7. Gullapalli, V.: A stochastic reinforcement learning algorithm for learning real-valued functions. Neural Networks 3, 671–692 (1990)
8. Martius, G.: Goal-Oriented Control of Self-Organizing Behavior in Autonomous Robots. PhD thesis, Göttingen University (2010)
9. Martius, G., Herrmann, J.M.: Tipping the scales: Guidance and intrinsically motivated behavior. In: Proc. of Europ. Conf. on Artificial Life (2011)
10. Martius, G., Herrmann, J.M., Der, R.: Guided Self-Organisation for Autonomous Robot Development. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 766–775. Springer, Heidelberg (2007)
11. Martius, G., Hesse, F., Güttler, F., Der, R.: Lpzrobots: A free and powerful robot simulator (2011), `robot.informatik.uni-leipzig.de`
12. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning 3, 9–44 (1988)
13. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998); A Bradford Book
14. Wiener, N.: Cybernetics or Control and Communication in the Animal and the Machine. Hermann, Paris (1948)