Design of a Control Architecture for Habit Learning in Robots

Erwan Renaudo[1,2], Benoît Girard[1,2], Raja Chatila[1,2], and Mehdi Khamassi[1,2]

[1] Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, Institut des Systèmes Intelligents et de Robotique, F-75005, Paris, France
[2] CNRS, UMR 7222, Institut des Systèmes Intelligents et de Robotique, F-75005, Paris, France

Abstract. Researches in psychology and neuroscience have identified multiple decision systems in mammals, enabling control of behavior to shift with training and familiarity of the environment from a goal-directed system to a habitual system. The former relies on the explicit estimation of future consequences of actions through planning towards a particular goal, which makes decision time longer but produces rapid adaptation to changes in the environment. The latter learns to associate values to particular stimulus-response associations, leading to quick reactive decision-making but slow relearning in response to environmental changes. Computational neuroscience models have formalized this as a coordination of model-based and model-free reinforcement learning. From this inspiration we hypothesize that it could enable robots to learn habits, detect when these habits are appropriate and thus avoid long and costly computations of the planning system. We illustrate this in a simple repetitive cube-pushing task on a conveyor belt, where a speed-accuracy trade-off is required. We show that the two systems have complementary advantages in these tasks, which can be combined for performance improvement.

1 Keywords

Adaptive Behaviour • Habit Learning • Reinforcement Learning • Robotic Architecture

2 Introduction

Researches in the field of instrumental conditioning in psychology have shown that rodents learning to press a lever in order to get food progressively shift from a goal-directed decision system to a habitual system [7,8]. After moderate training, devaluation of the outcome (e.g. pairing it with illness) leads the animal to quickly stop pressing the lever. In contrast, after extensive training the animal perseveres with pressing the lever even after outcome devaluation - hence "habit" [1,21]. This has been hypothesized to enable the animal to avoid slow and costly decision-making through planning by shifting to reactive decision-making when the stability of the environment makes habits reliable, a capacity which is shared with humans and other mammals [2].

In contrast, current robots are still rarely equipped with efficient online learning abilities and mostly rely on a single planning decision-making system, thus not providing alternative solutions to motion planning in situations where such strategy is limited [17]. Indeed the planning strategy can be approximative when coping with uncertainties, *e.g.* when there is perceptual aliasing [4], and can also require high computational costs and long time to propagate possible trajectories through internal representations [10]. We have previously shown that taking inspiration from the way rodents shift between different navigation strategies – a capacity which has been shown to be analogous to the shifts between goal-directed and habitual decision systems [14] – can be applied to a robotic platform to enable to automatically exploit the advantages of each strategy [4,3]. However, these experiments only involved navigation behaviors from one location to another. To our knowledge, no application has yet been made of the coordination of goal-directed and habitual systems to robotic tasks.

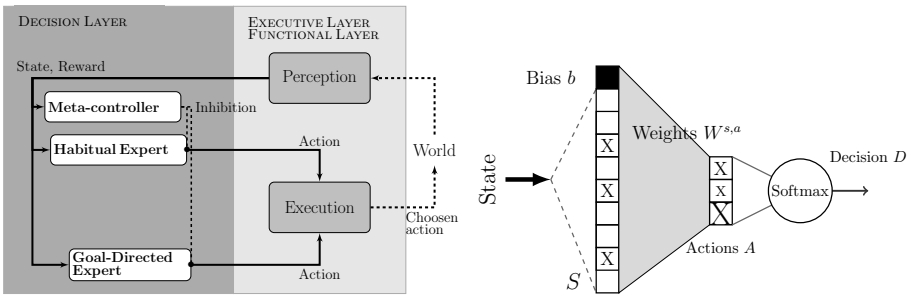
In this work, we illustrate the application of a decision architecture combining a goal-directed expert with a habitual one to a simple task where a simulated robot have to learn to repeat the less costly sequence of actions to push a series of cubes arriving in front of him on a conveyor belt. We build our algorithm on computational neuroscience models which have shown that combining model-based and model-free reinforcement learning can accurately reproduce properties of the competition between goal-directed and habitual systems [5,13,9]. In these models, the goal-directed system is modelled with model-based reinforcement learning in the sense that the system plans sequences of actions towards a particular goal by using the transition and reward functions. In parallel, the model-free reinforcement learning progressively learns by trial-and-error the Q-values associated to different state-action couples. The criterion for switching from one system to the other is based on the measure of uncertainty in the model-free system: the less variance there is in the Q-values, the more reliable the model-free habitual system is considered and the more likely it will control the behavior of the simulated agents.

In contrast to these previous computational neuroscience models, we do not a priori give the transition and reward functions (i.e. the considered model of the task) to the algorithm but rather make it learn it automatically by observing experienced transitions and rewards. Moreover, we arbitrate without bias between systems, as the selection of each one is random and equiprobable. The task that we simulate requiring a certain balance between speed and accuracy so as not to skip some cubes coming on the conveyor belt, our simulations show that the two systems have complementary advantages that can be combined for a highest performance. In a first series of simulations where the systems are controlling individually the agent, we characterize their performances in a constant belt velocity and constant distance between cubes setup and when the belt velocity is changed during the simulation. We show that each system is performing differently to these conditions as the model-free is more efficient than the model-based to exploit the stability in the environment, but the model-based adapts quickly to condition changes in the environment. We then show how combining

the two systems and switching control among them, even with a basic rule, can improve the robot policy and gives it the ability to perform well both in a stable environment and during transitional phases to another stable setup, with the same architecture.

3 Materials and Methods

3.1 Global Architecture



(a) The inner structure of the Decision Layer and its connection with the Executive and Functional Layers.

(b) The Habitual Expert neural network. Black X size is proportional to activity in the network.

Fig. 1: Robotic organisation of modules and Habitual Expert structure

Our Decision Layer [10] consists of two Experts that learn a policy and a Meta-Controller that supervises the Experts' performance (Fig. 1a). The Flexible Expert is a Model-Based Reinforcement Learning agent and the Habitual Expert is a Model Free reinforcement learning agent [19].

These modules receive the current State $S \in \mathcal{S}$ from a Perception Module and choose their actions in a set $\mathcal{A}$. Each Expert decides, from the current State and their knowledge, which action to take. In parallel, the Meta-Controller decides which Expert is the most efficient in the current State and allows it to send its action choice to be executed.

3.2 Habitual Expert

The Habitual Expert (MF) is implemented as a 1-layer neural network. It learns directly the relevant state-action policy without an internal representation of transitions between states of the world (hence the term Model Free). Propagating the values from input S (and bias b) to action output A is computationally cheap, but learning the whole policy is long: only the experienced state-action value is updated. Learning a new policy to adapt to a new environment configuration is longer than just learning the first policy so this expert is reluctant to changes.

$$A_t(i) = W_t \cdot S_t + b_t(i) \ . \tag{1}$$

The connection weights W_t that learn the State-Action association are updated according to a Qlearning rule [20]: the connection between input neurons encoding the previous State S_{t-1} and the output neuron of the action done is modified with an amount depending on the reward R obtained.

$$\delta = R_t(s_{t-1}, a_{t-1}) + \gamma_{MF} \cdot max_a \left(W_{t-1}(a) \cdot S_t\right) - \left(W_{t-1}(a_{t-1}) \cdot S_{t-1}\right) \ . \quad (2)$$

$$W_t(a_{t-1}) = W_{t-1}(a_{t-1}) + \alpha_{MF} \cdot \delta \ . \quad (3)$$

α_{MF} : learning rate, γ_{MF} : decay factor.

Each action activity is interpreted as the probability $P(A(i))$ of taking action $A(i)$, using a Softmax rule (4). The decision is taken stochastically in the resulting distribution (τ_{MF} : temperature.).

$$P_t(A_t(i)) = \frac{\exp\left(\frac{A_t(i)}{\tau_{MF}}\right)}{\sum_j \exp\left(\frac{A_t(j)}{\tau_{MF}}\right)} \ . \quad (4)$$

3.3 Flexible Expert

The Flexible Expert (MB) is a Model Based Reinforcement Learning agent. It learns a model of the *Transition* and *Reward* functions of the task. The former is a *cyclic graph* of States connected by Actions, the latter a *table* of (State, Action) and Reward association. Decisions are taken based on these representations of the world. As the problem topology is modeled, a change experienced in the environment (ie. a transition leads to a new state) can quickly be handled by updating the model, allowing the next decision to be adapted to the changes.

The Reward function is learned from the experienced transition and is directly the instant reward obtained $R_t(S, A) = R_t$. The Transition function is progressively learned according to (5). The probability T of experienced transition $S \xrightarrow{A} S'$ is updated at learning rate α_{MB}.

$$T_t(S, A, S') = T_{t-1}(S, A, S') + \alpha_{MB} \cdot \left(1 - T_{t-1}(S, A, S')\right) \ . \quad (5)$$

Planning with the models consists in computing the Quality Q(S,A) of performing action A in the given state S. It is done iteratively by propagating the known rewards and refining the estimated Quality value according to the Transition function until convergence (γ_{MB} : decay factor):

$$Q_t(s, a) = max\left(R_t(s, a), (\gamma_{MB} \cdot \sum_{s'} T_{t-1}(s, a, s') \cdot \max_{a'} Q_t(s', a'))\right) \ . \quad (6)$$

The Decision is also taken with the *softmax rule* (cf. Eq. (4)).

The drawback of such a method comes from the increasing size of the transition model. Planning becomes more and more time consuming and the Expert is

less and less reactive. As the environment evolves, even in a predictable way, the action decided from the perceived State at S_{t-1} may be irrelevant when acting in State S_t. To improve the Flexible Expert performance and keep a manageable model while dimensionality increases, the following features are implemented :

1. Planning in the graph is bounded in time : if planning is longer than a certain time chosen in agreement with the task dynamics, the computation of Quality is stopped and the approximated values are used for decision, as it is more important to be reactive enough than having accurate values in this task.
2. As the best policy is learnt, fewer and fewer states are visited, producing a peaked distribution V_S of states visits. The allocated computation time being limited, planning should only consider the most visited states. These states are hypothetized to be the most interesting for the Expert, as the policy focuses on a subset of all experienced states. A subgraph of the N most visited states is extracted to have their Q-values computed as a priority. To have a relevant value for N, we compute the entropy of the V_S distribution, getting a measure of the model organisation :

$$H(V_S) = -\sum_{i \in S} P(i) \cdot log_2\left(P(i)\right) \text{ with } P(i) = \frac{Card(V_{S_i})}{Card(V_S)} \ . \tag{7}$$

We compare this measure to the maximal entropy of the model, deducing a ratio R_c of the compressibility of the State distribution representation :

$$R_c = \frac{H(V_S)}{H_{max}(V_S)} \text{ with } H_{max}(V_S) = log_2|\mathcal{S}| \ . \tag{8}$$

This method guides the planification to the most visited states. The drawback is that it may erroneously limit the use of the model during early states of learning where the number of states is small but the distribution already presents a contrasted shape. In this case, planning in the full graph is still possible at reasonable cost. To avoid this behaviour, R_c is transformed into a Ratio R_n - depending on the known number of nodes - given the following function (9).

$$R_n = (1 - \omega) + \omega \cdot R_c \text{ with weight } \omega = \frac{1}{1 + e^{-\sigma|\mathcal{S}|}} \ . \tag{9}$$

The final number of states to plan on is a proportion of the number of known states $|\mathcal{S}|$:

$$N = R_n \cdot |\mathcal{S}| \ . \tag{10}$$

3.4 Meta-Controller

The Meta-Controller gives the control to one of the Experts given a criterion. It allows only one of the Experts to send its decision to the Execution Layer. It also sends back the decision to both Experts, such that they can update their

knowledge about its relevance in the current state according to the feedback, and cooperate in learning the best policy. The criterion considered in this work is an equiprobable random selection of each Expert, as a proof of concept of the interest of combining the two.

4 Results

4.1 Experiment Description

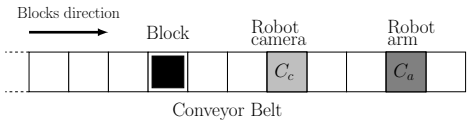


Fig. 2: The experimental setup : a discrete conveyor belt is carrying blocks in front of the robot. The robot's camera points at space C_c and its arm can reach the space in C_a. Blocks are going from left to right such that a block can be first seen and then touched.

We evaluated our Architecture performance in the simulation of a simple task of block pushing. The system has been implemented using the ROS middleware [18]. Our simulated robot is facing a conveyor belt on which are placed blocks. These blocks are characterized by their velocity (BS) and the distance between two blocks (inter-block distance, or IBD). These simulation parameters may be constant or evolve during the experiment, leading to four different cases. In this work, we focused on :

1. Regular case : inter-block distance is constant, speed of blocks is constant.
2. Speed Shift case : IBD is constant, BS changes during experiment.

In our setup, acting is required to update the perception (see Sect. 4.2 for Perception Module description). The robot has three available actions :

1. **Do nothing (DN) :** this action doesn't modify the environment nor bring perceptual information. It is a waiting action with no cost ($R_t = 0$) when executed.
2. **Look Cam (LC) :** this action doesn't modify the environment but updates the view modality about the presence of a block in C_c. It has a cost of $R_t = -0.03$.
3. **Push Arm (PA) :** this action can modify the environment : if a block is in C_a and PA is done, it is removed from the belt. The contact modality is updated about the perceived block. The action costs $R_t = -0.03$ but brings a positive reward when a block is pushed for a final reward of $R_t = 0.97$).

4.2 Perception Module

The Perception Module (Fig. 3) transforms Perceptions into States. Our simulated robot is equipped with a visual block detector simulated camera (signal

p^{bs}) and a tactile binary sensor on its arm (signal p^{bt}). When the corresponding action is selected (LookCam for p^{bs} and PushArm for (p^{bt}), these informations update memories where each element is one step further in the past. Each modality has its own memory where older block perceptions are recorded. Memories (M^{bs}, M^{bt}) have a finite length (8 elements) such that the system only considers closest perceptions. Each configuration of both memories defines a unique State, used by the Experts.

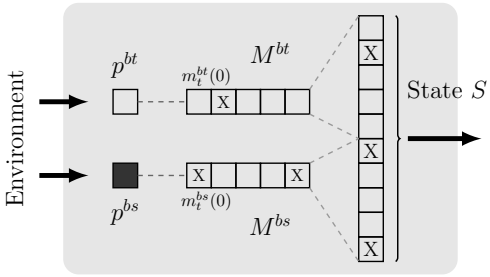


Fig. 3: Perception module for our task. The X corresponds to a memorized block. The system visually perceives a block and updates the corresponding memory. *bs* and *bt* stand for *block seen* and *block touched*

The perceptive input are binary and their perceptions are determined according to (11).

$$p^{bt} = C_a \cdot \text{PushArm}, \, p^{bs} = C_c \cdot \text{LookCam} \ . \tag{11}$$

Memories can evolve in two ways : if enough time has elapsed (State Max Duration = 0.1s here) or when a new perception brings information. In both cases, all elements from the memory are shifted to the next timestep. Information that exceeds memory length is forgotten. The first memory element is populated with the relevant perceptive data.

$$\begin{cases} m_t^{bs,bt}(i) = m_{t-1}^{bs,bt}(i-1) \ \forall i \in |M| \ . \\ m_t^{bs,bt}(0) = p^{bs,bt} \ . \end{cases} \tag{12}$$

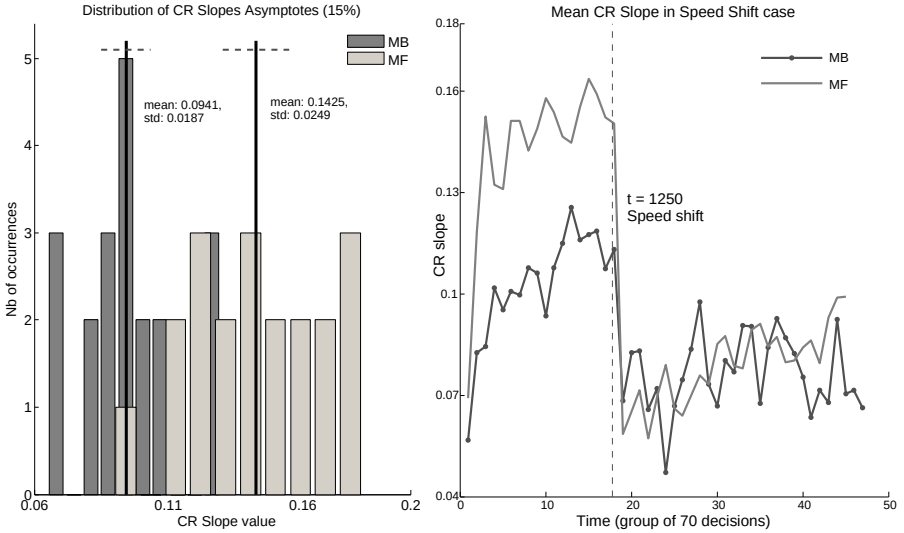
4.3 Parameters search

In the following, we consider : $BS = 8$ spaces/s (optimal policy : DN-DN-DN-DN-PA), $IBD = 4$ spaces/block.

We first searched for the best parametrization for both Experts controlling individually the robot, in the Regular case. An Expert is performant if it maximises the obtained Cumulative Reward (CR) and minimizes the standard deviation of CR over runs. We tested for each Expert the combination of 3 to 5 values (for MB and MF : $\alpha_{MB,MF} \in \{0.01, 0.05, 0.1, 0.5, 0.9\}, \gamma_{MB,MF} \in \{0.5, 0.98, 0.9999\}, \tau_{MB,MF} \in \{0.05, 0.1, 0.5, 0.9\}$ plus $\tau_{MB} = 0.01$).

For the best solutions, we favor the most rewarding in mean and then the less varying. We choose $\alpha_{MB} = 0.5, \gamma_{MB} = 0.5, \tau_{MB} = 0.1$ and $\alpha_{MF} = 0.1, \gamma_{MF} = 0.9999, \tau_{MF} = 0.05$.

4.4 Individual experts performances



(a) Regular case (20 runs) (b) Speed shift case (10 runs)

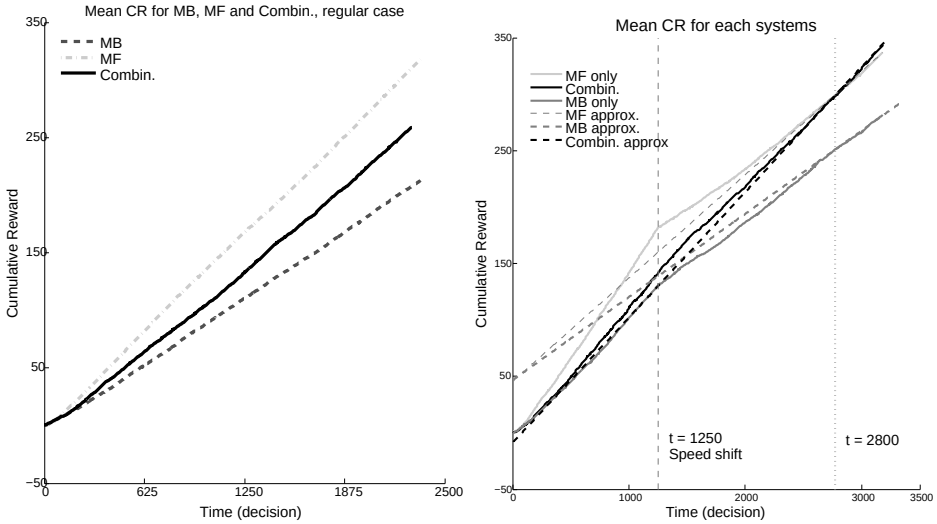
Fig. 4: Policy evaluations. (4a) histogram of approximated slopes. Solid lines are means of CR slope for each Expert, dashed lines the standard deviation (4b) Mean CR slope over time. The slope is approximated every 70 decisions.

Each Expert is tested individually in the Regular and Speed Shift cases (simulation parameters : see Sect. 4.3 ; in Speed Shift case, we change BS to 13.2 spaces/s at 1250 decisions, which correspond to an optimal DN-DN-PA policy). The Cumulative Reward is linearly approximated from its 15% last values to evaluate the discovered policy. For the Speed shift case, we also approximate CR before the speed shift (on the same duration) and compare it to the first approximation to evaluate the sensitivity of the policy to speed shift. The slope of these approximations measures the quality of the policy as it depends on the obtained rewards. From figure 4a, we observe that the MF discovers and follows better policies in mean but tends to be more exploring than the MB with a larger deviation in slopes values. In the Regular case, the MF is more relevant than the MB to obtain the best performance as possible. This is due to the low cost and high precision of the Q-values acquired by the MF. In contrast, the MB learns a model of the task whose number of states rapidly grows, making the planning process slow, costly and relying on approximations of action values.

In the Speed Shift case, we observe a break in Cumulative reward for the MF. Figure 4b shows that in mean, the MB performance is less sensitive to the change than the MF : the environmental change induces a loss that is more than twice higher in the MF than in the MB. After the shift and until the end of simulation, both MB and MF are performing similarly. This behavior can be explained by the long time required by the MF to relearn the Q-values of a new efficient action sequence, what doesn't happen in the given time. In contrast, a single exposure of the MB to the new sequence of events imposed by the speed shift enables it to change its model of the task and thus to plan a new sequence of actions, but it still suffer from the approximation of action values to find a better policy.

Both Experts exhibit a complementary role : while the MF is best suited to optimize the policy in stable conditions, the MB can better handle transient phases following environmental changes.

4.5 Combination of Experts



(a) Regular case (20 runs) (b) Speed shift case (10 runs)

Fig. 5: Mean Cumulative Reward obtained from individual Experts and their Combination (solid line).

The whole architecture (MB+MF, supervised by Meta-Controller) is then tested on both cases, with the same setup. In the Regular case, figure 5a shows that the strategy of selecting stochastically each Expert improves the mean performance of the robot compared to using only the MB. On the other hand, as the Experts are chosen randomly, the robot is not relying only on the MF, which is the most efficient strategy in the Regular case. This explains that the MB+MF

performance is still worse than the MF only. In the Speed Shift case, figure 5b shows that the change in the environment doesn't affect significantly the robot's performance, as it benefits from the MB ability to quickly replan an adapted policy. The Combination of Experts robustness to changes compensates for the advantage gained by the MF before the shift. At the end of the simulation, the MF hasn't found a policy that is at least as good as before the shift (though the task allows a higher rate of reward, as there are more blocks to push on).

5 Discussion

This work presented the decision layer of a robotic control architecture able to learn habits, taking inspiration from computational neuroscience models [13] and multiple reinforcement learning systems applied to navigation [9,4]. The model has two different Experts, or strategies, one habitual – that learns State-Action association, and make quick decisions, but slowly adapts its policy when the environment changes – and one flexible – that maintains a representation of the environment and the task, can adapt quickly but is slow in deciding as it evaluates action outcomes. These strategies are selected depending on an arbitration criterion by a Meta-Controller. The criterion used in this work is an random equiprobable selection of Experts, as a proof-of-concept of the interest of combining the two.

We first highlighted each Expert properties in a Regular and a Speed Shift cases. We showed that, as expected, the Habitual Expert learns better policies than the Flexible Expert when the environment is stable, but a transient phase like a shift in belt speed, making the policy less appropriate, will result in a long lower performance period. The learnt Flexible Expert policies are less performant in mean as the computation time constraint and the focused planning lead to less precise Q-values when the number of states grows, and a sub-optimal policy. On the other hand, updating its model allows the Flexible Expert to be less affected by the speed shift. We then showed that a random selection of each Experts is able to benefit from the shift robustness of the Flexible Expert while the rewarding policies from the Habitual Expert improve the global performance of the robot.

These results show that the multiple reinforcement learning systems approach is relevant to handle complex environments that can evolve during the robot operating period. The combination at same level of MB and MF can improve the robot autonomy provided that the MB is designed to be reactive to the environment dynamics. It has to be able to decide in parallel with the MF in order to remain useful for control. Indeed, a classical task in neuroscience is usually modelled by a Markov Decision Process with few states and actions (e.g. intrumental task of pressing a lever and entering a magazine to get food [6,13,16]) but the dimensionality is much higher when reinforcement learning is applied to robotics [12], as states are discretized from robot's perceptions. In our not so simple task, we end up with several hundreds of states and we need to bound the computation time and focus planning on the hypothesized most interesting states. This

is a well-known issue of applying Reinforcement Learning to robotics [15] and justifies the need for a mechanism that manages the known states within the MB system, a proposition which has recently been applied to Computational Neuroscience models [11]. In this work, we first tested an exponential forgetting mechanism on the transition model to remove unvisited paths. As this mechanism doesn't strongly affect performance and planning time is still increasing with the growth of states, we switched to the time constraint and focused planning mechanism (described in Sect. 3.3). The increase in performance suggests that planning with a complex model requires a budgeted approach that only considers the relevant sub-model, instead of pruning parts of the model related to irrelevant old experiences. Our Experts' parametrization has been optimized for the regular case. In a first approach, we choose the parameters to be constant but in this task, the MB should be exploring enough such that the agent gets out of the initial attractor state and then exploit its model to quickly adapt to environmental changes. Here, both the Experts have an exploiting temperature, and the lower adaptability of the MF comes from the way it learns (updating only the experienced actions).

This work also generalizes the concepts from [4] for the control of robots. The multiple reinforcement learning systems approach can be applied not only for navigation but also on a wider variety of tasks, provided that the robot is able to perform the relevant actions. As our system can rely on the Habitual Expert, our architecture can benefit from its properties of being quick to decide the next action when the task is stationary. On the other hand, our Flexible Expert can be compared to the robotic decision-making systems, that are based mostly on planning algorithms that use a representation of the world [17]. The latter usually rely on a provided representation whereas our Flexible Expert learns its model and updates it according to changes in the task. This enhances the behavioral adaptability of the robot in non-stationary environments. We need to further investigate the arbitration criterion between Experts to get the optimal alternations and benefit from the whole architecture.

Acknowledgements

This work has been funded by a DGA (French National Defence Agency) scholarship (ER), by the Project HABOT from Ville de Paris and by French Agence Nationale de la Recherche ROBOERGOSUM project under reference ANR-12-CORD-0030.

References

1. B. W. Balleine and A. Dickinson. Goal-directed instrumental action: contingency and incentive learning and their cortical substrates. *Neuropharmacology*, 37:407–419, 1998.
2. B. W. Balleine and J. P. O'Doherty. Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology*, 35:48–69, 2010.

3. K. Caluwaerts, A. Favre-Félix, M. Staffa, S. N'Guyen, C. Grand, B. Girard, and M. Khamassi. Neuro-inspired navigation strategies shifting for robots: Integration of a multiple landmark taxon strategy. In T.J. et al. Prescott, editor, *Living Machines 2012, LNAI*, volume 7375/2012, pages 62–73. 2012.
4. K. Caluwaerts, M. Staffa, S. N'Guyen, C. Grand, L. Dollé, A. Favre-Félix, B. Girard, and M. Khamassi. A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration & Biomimetics*, 2012.
5. N.D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, 2005.
6. Amir Dezfouli and Bernard W. Balleine. Habits, action sequences and reinforcement learning. *European Journal of Neuroscience*, 35(7):1036–1051, 2012.
7. A. Dickinson. *Contemporary animal learning theory.* Cambridge: Cambridge University Press, 1980.
8. A. Dickinson. Actions and habits: The development of behavioral autonomy. *Philosophical Transactions of the Royal Society (London)*, 308:67 – 78, 1985 1985.
9. L. Dollé, D. Sheynikhovich, B. Girard, R. Chavarriaga, and A. Guillot. Path planning versus cue responding: a bioinspired model of switching between navigation strategies. *Biological Cybernetics*, 103(4):299–317, 2010.
10. E. Gat. On three-layer architectures. In *Artificial Intelligence and Mobile Robots.* MIT Press, 1998.
11. Q.J. Huys, N. Eshel, E. O'Nions, L. Sheridan, P. Dayan, and J.P. Roiser. Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Computational Biology*, 8(3), 2012.
12. L.P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
13. M. Keramati, A. Dezfouli, and P. Piray. Speed/accuracy trade-off between the habitual and goal-directed processes. *PLoS Computational Biology*, 7(5):1–25, 2011.
14. M. Khamassi and M.D. Humphries. Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. *Frontiers in Behavioral Neuroscience*, 6:79, 2012.
15. J Kober, D. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. (11):1238–1274, 2013.
16. F. Lesaint, O. Sigaud, S. B. Flagel, T. E. Robinson, and M. Khamassi. Modelling Individual Differences in the Form of Pavlovian Conditioned Approach Responses: A Dual Learning Systems Approach with Factored Representations. *PLoS Comput Biol*, 10(2), February 2014.
17. J. Minguez, F. Lamiraux, and J.P. Laumond. Motion planning and obstacle avoidance. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics.*, pages 827–852. Springer-Verlag, 2008.
18. M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
19. Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.
20. C. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, 1989.
21. H. H. Yin, S. B. Ostlund, and B. W. Balleine. Reward-guided learning beyond dopamine in the nucleus accumbens: the integrative functions of cortico-basal ganglia networks. *Eur J Neurosci*, 28:1437–1448, 2008.