Evolutionary Learning Outperforms Reinforcement Learning on Non-Markovian Tasks

G. de Croon, M.F. van Dartel, and E.O. Postma

IKAT, Universiteit Maastricht, P.O. Box 616, 6200 MD, Maastricht, The Netherlands
{g.decroon, mf.vandartel, postma}@cs.unimaas.nl,
http://www.cs.unimaas.nl

Abstract. Artificial agents are often trained to perform non-Markovian tasks, i.e., tasks in which the sensory inputs can be ambiguous. Agents typically learn how to perform such tasks using either reinforcement learning (RL) or evolutionary learning (EL). In this paper, we empirically demonstrate that these learning methods result in different levels of performance when applied to a non-Markovian task: the Active Categorical Perception (ACP) task. In the ACP-task, the proportion of ambiguous sensor states can be varied. EL outperforms RL for all tested proportions of ambiguous states. In addition, we show that the relative performance difference between RL and EL increases with the proportion of ambiguous sensor states. We argue that the cause of this increasing performance difference is that in RL the learned policy consists of those state-action pairs that individually have the highest estimated values, while the performance of a policy for a non-Markovian task highly depends on the *combination* of state-action pairs selected.

1 Introduction

Artificial agents often have to perform tasks in which the sensory inputs can be ambiguous, referred to as non-Markovian tasks [8, 1]. Agents typically learn how to perform such tasks with either reinforcement learning (RL) or evolutionary learning (EL). RL and EL differ fundamentally in the manner in which they search in the policy space, i.e., the space of possible mappings from states to actions. RL searches for the optimal policy by learning a value function of states or state-action pairs. Learning a value function is more difficult in a non-Markovian task than in a Markovian task, since it is hard to estimate the value of an ambiguous sensor state [1, 9]. In contrast to RL, EL searches in the policy space directly, by selecting and evaluating complete policies. Therefore, EL does not depend on how well the values of ambiguous states can be estimated, but on the properties of the policy space in which it searches. This difference between value function search and direct policy search leads to the conjecture that EL performs better on non-Markovian tasks than RL does [1, 12, 5]. If the estimation of the value of ambiguous states is the problem for RL, EL should be better able to cope with a larger proportion of ambiguous states.

In this paper we investigate empirically whether this conjecture is valid. Therefore, we formulate the following two research questions: (1) *Do reinforcement learning and evolutionary learning result in different levels of performance when applied to a non-Markovian task?*, and (2) *Is there a relation between the proportion of ambiguous states and the performance difference between reinforcement learning and evolutionary learning?*

To answer these research questions, we use a revised version of the 'Active Categorical Perception' (ACP) task (described in [11]). We adopt this task, since it allows a gradual increase in the proportion of ambiguous states. Although the task is non-Markovian, in [11] it has been shown that memory-less, deterministic policies can be rather successful in solving the task. Since these types of policies are the ones that have been most extensively studied in the literature (e.g., [6, 11]), we focus on such policies.

The remainder of the paper is organised as follows. In Sect. 2, we present the revised version of the ACP-task, which we employed to compare the two learning methods. We describe the experiments in Sect. 3. The results are reported in Sect. 4. In Sect. 5 we analyse the experimental results. We discuss the outcome of the analysis in Sect. 6. Finally, we draw our conclusions in Sect. 7.

2 Active Categorical Perception Task

To compare RL and EL, we apply them to a revised version of the ACP-task as described in [11]. In the active categorical perception task, an agent has to categorise falling objects by catching or avoiding them. Below, we provide an overview of the revised version of the task.

2.1 Overview of the ACP-task

In the ACP-task [11], an agent has to catch small objects and avoid large objects. By doing so, an agent exhibits its ability to categorise. The environment in which an agent acts is a two-dimensional grid of 20 x 10. The objects and agents are allowed to move through the left and right boundaries of the environment, and to re-appear at the opposite side of the environment. An 'epoch' consists of ten discrete time steps. At the first time step an object is placed somewhere in the top row of the grid. The object falls down one row at each time step, and hits the floor of the grid at the last time step. Objects always fall with a horizontal velocity of two grid cells leftward or rightward. The direction of movement of the objects is consistent throughout an epoch. Small objects occupy two grid cells, while large objects occupy four grid cells. The agent is restricted to movements in the bottom row of the environment. Figure 1 illustrates three consecutive simulation time steps of an epoch (denoted by $t = 5$ to $t = 7$). In the figure, a large object (represented by four filled grid cells) falls rightward. The four circles in the bottom row of each grid represent the sensors of the agent; they are activated (gray circles) by the presence of an object in the same column. In the figure, the agent moves four grid cells leftward at each time-step.

The movement of the agent depends on the activation of the sensors and the agent's policy. In our experiments, a policy is represented by a state-action

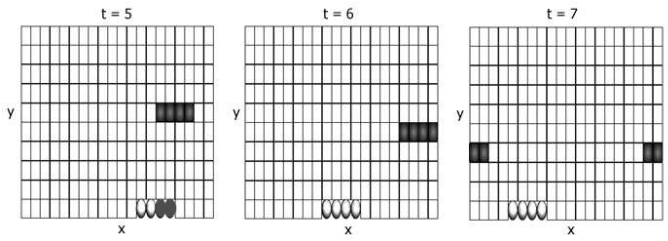

Fig. 1. Movement of an agent and object over three consecutive simulation time steps (from $t = 5$ to $t = 7$). Filled grid cells represent the object and circles represent the active (gray) and inactive (white) sensors of the agent.

table that assigns a value to all possible actions for all possible sensor states. The state-action table determines the size and direction of the next step taken by the agent, i.e., the number of grid cells moved to the left (negative action) or right (positive action), ranging from -10 to 10. If the action is 0, the agent does not move. A movement of the agent leads to a new position of the sensor array and, consequently, to a new sensor state. The sensor array of the agent consists of two types of sensors: functional sensors and blind sensors. Each agent has s functional sensors. A variable number of blind sensors b replace functional sensors s in the sensor array; they determine the level of perceptual ambiguity (see Subsection 2.2). In the experiments described below, we optimise agents with all possible configurations of functional sensors and blind sensors. A functional sensor is active if an object is present in its column, and inactive otherwise. We note that the agent's sensor state does not contain any information regarding its own position or its distance to the object.

The behaviour of an agent is evaluated when the object reaches the bottom row of the grid. An object is caught by the agent iff $ca - co \leq d$ (modulo the boundaries), with ca representing the center of the agent, co the center of the object, and d a number of grid cells; an object is avoided iff $ca - co > d$ (modulo the boundaries). Following the setup in [11], we determine $d = 4.5$. Since the center is a size-independent measure, there is an equal probability to avoid or catch an object from both classes. The performance of a policy is defined as the percentage of objects that are correctly classified. Therefore, if both types of objects are encountered an equal number of times, the expected performance of random behaviour is equal to 50%. Since the environment is deterministic and there are a small finite number of possible distinct epochs, we evaluate the performance of an agent by executing it for all possible starting positions and moving directions of both types of objects. We refer to an execution of all distinct epochs as an 'episode'.

2.2 Ambiguous Sensor States

We define an 'ambiguous sensor state' as a sensor state that does not provide any information on the type of object present in the environment. For example, if the agent consists of four active sensors ($s = 4$) and only the left sensor is active, this sensor state might be due to either a big object or a small object. The number

of ambiguous sensor states can be changed by altering the sensor configuration of the agent. For instance, in the case that all sensors are functional, there are nine possible sensor states of which five are ambiguous (left in Fig. 2). In the case of a blind leftmost sensor only six sensor states can occur, of which five are ambiguous (right in Fig. 2).

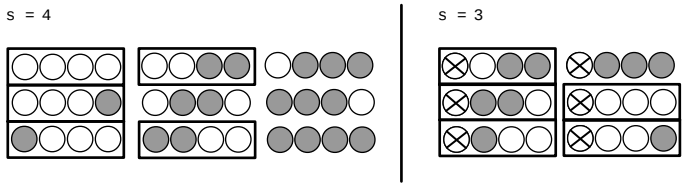

Fig. 2. Left of the line, all possible sensor states are shown when all four sensors are functional. Right of the line, all possible sensor states are shown when the leftmost sensor is blind. The circles represent the active (gray), inactive (white), and blind (crossed) sensors of the agent. Ambiguous sensor states are indicated with a box. Examples of unambiguous states left of the line are the states with three or four active sensors (large object) and with two active middle sensors (small object).

In the experiments described below, we apply both RL and EL to the task with all sensor configurations and measure the performance differences. A specific sensor configuration does not imply that the agent encounters a specific proportion of ambiguous states, since the proportion of ambiguous states actually encountered also depends on the agent's behaviour [6]. However, it is possible to estimate the proportion with a behaviour-independent measure: the 'task ambiguity'. We define the task-ambiguity as the proportion of object positions that give rise to an ambiguous sensor state. For instance, for four functional sensors the task ambiguity is 90%, since 90% of the possible object positions results in one of the ambiguous sensor states depicted on the left in Fig. 2.

3 Experiments

To test how well RL and EL perform on a non-Markovian task, we apply both learning methods to the categorisation task described in Sect. 2. In addition, we employ random search as a reference point. In the next subsections we describe the applied learning methods: reinforcement learning (Subsection 3.1), evolutionary learning (Subsection 3.2), and random search (Subsection 3.3). As described below, for each learning method we optimise its parameter settings for the task with four functional sensors. The optimised parameter settings are then employed in experiments on all distinct sensor configurations.

3.1 Reinforcement Learning (RL)

There have been several studies that analyse the performance of RL-methods on non-Markovian problems [9, 4, 3]. In [9] Monte Carlo (MC) is recommended to learn non-Markovian tasks, while in [4, 3] it is argued that time differential learning with eligibility traces (SARSA(λ)) is a better learning method for such tasks.

We implement MC, SARSA(λ), and include Q-learning, since it is a widely-used RL-method (for an explanation of these methods, see [10]).

We optimise the parameter settings of each learning method on the task with four functional sensors as follows. For every parameter setting we perform 30 different runs of the learning algorithm, each consisting of 5000 episodes. We first vary all relevant parameters from 0 to 1.0 with steps of 0.1. Around the optimum, these rates are varied on a finer scale with steps of 0.02. If the optimum is at the end of the scale, experiments are run with steps of 0.005. The relevant parameters are: the exploration rate ϵ (all methods), the training rate α (SARSA(λ) and Q-learning), the discount factor γ (SARSA(λ) and Q-learning), and the eligibility parameter λ (SARSA(λ)). Note that exploration is not applied to the last run of an episode, so that the learned policy can be evaluated. In [3] it is argued that the exploration rate should decrease over time. Therefore, we also employ all RL-methods with a decreasing exploration; as in [3], the initial exploration rate ϵ is set to 0.20, which decreases linearly during learning, so that it reaches 0 at the 1000^{th} episode. After optimising the parameter settings, we apply all RL-methods to the task with all possible sensor configurations, performing 100 runs per configuration.

3.2 Evolutionary Learning (EL)

In our EL-algorithm, a population is evolved for 100 generations and consists of 50 individuals, i.e., policies. The genome of an individual represents a state-action table by a vector of doubles in the range $[0, 1]$. During an episode, the agent chooses the action that has the highest value in the genome for the current state. After evaluating all individuals in the population, the best 50% of them are selected to produce the next population. Each selected individual has two offspring in the new generation. To form one offspring, the genome of the individual has a probability of P_c that a one-point cross-over at a random location in the genome is performed with another selected individual. After a possible cross-over, each gene in the genome is mutated with probability P_m to a random value in the range $[0, 1]$. For the experiments of evolutionary learning we optimise the relevant parameters (P_c and P_m) in the same manner as the parameters of the RL-methods, and employ the optimised parameter settings in experiments on all sensor configurations.

3.3 Random Search (RS)

We compare both RL and EL to random search (RS) on all sensor configurations, as a reference point for their performance. Like EL, RS performs direct policy search. For RS, we perform 100 runs on all sensor configurations, one run consisting of evaluating 5000 random policies.

4 Results

Below, we first discuss the results of the experiments on the task with four functional sensors. Then, we compare all RL-methods with EL and RS on all possible sensor configurations.

Figure 3 shows the performance over time of all RL-methods and EL on the task with four functional sensors, averaged over all hundred experimental runs. For each method, we only show the results of the settings that achieved the highest mean performance. These settings are: for MC $\epsilon = 0.02$; for SARSA(λ) a decreasing ϵ, $\lambda = 0.14$, $\gamma = 0.8$, and $\alpha = 0.005$; for Q-learning $\epsilon = 0.01$, $\gamma = 1.0$, and $\alpha = 0.005$; for EL $P_c = 0.20$ and $P_m = 0.06$. The main observation we can make from Fig. 3 is that, although both MC and Q-learning initially perform better than EL, EL eventually achieves a higher performance than any of the RL-methods. Note that the increase in performance of SARSA(λ) almost comes to a halt at the 1000^{th} run, the cut-off point of the exploration. Other settings of the cut-off point (3000, 4000 episodes) yielded similar results, but attained a lower performance.

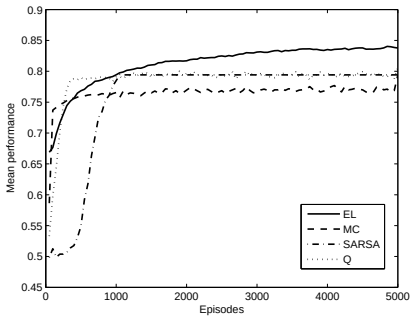

Fig. 3. Rewards over time, expressed in episodes, for all RL-methods and EL.

Table 1 shows the results for all sensor configurations ('SC', first column), where 'x' stands for a blind sensor and 'o' for a functional sensor. Since in one episode all objects are placed at all positions for both moving directions, mirrored sensor configurations give equal results and are not included in the table. The second column of the table contains the task ambiguity ('TA'). The five columns that follow, contain the average performance and standard deviation for all learning methods. The significance of the performance differences between the various methods has been tested with a bootstrap method [2] ($p < 0.05$). The last five columns contain the maximal performances of all learning methods. In Table 1, the highest mean performances are in bold text. We can make four observations from Table 1. The first observation is that EL significantly outperforms all RL-methods on all sensor configurations. Secondly, Q-learning significantly outperforms the other RL-methods on five sensor configurations (a '2' indicates that there is no significant performance difference between Q-learning and MC for four of the configurations). Thirdly, RS significantly outperforms all RL-methods from the third sensor configuration on. Finally, the table shows that EL achieves higher maximal performances than all other learning methods on the first two sensor configurations, and equal or higher maximal performances on all other sensor configurations.

Table 1. Mean performance, standard deviation, and maximal performance of all learning methods for all possible sensor configurations. A '1' indicates that the performance difference with EL is statistically not significant ($p < 0.05$), and a '2' indicates that the performance difference with Q-learning is not significant ($p < 0.05$).

SC	TA	EL	RS	MC	Q	SARSA	EL	RS	MC	Q	SARSA
oooo	90.0	**85.2(±1.9)**	77.9(±2.0)	79.4(±4.4)	80.6(±3.6)	79.4(±3.7)	88.75	83.75	86.25	86.25	86.25
oxoo	92.5	**83.5(±2.3)**	78.0^{2}(±2.5)	75.4(±5.1)	77.9(±4.8)	74.8(±4.5)	87.50	85.00	83.75	85.00	82.50
xooo	95.0	**80.4(±2.3)**	75.8(±2.1)	71.8(±4.7)	74.6(±5.5)	71.7(±4.7)	85.00	81.25	82.50	83.75	85.00
xoxo	95.0	**76.6(±1.5)**	75.5(±1.5)	67.2(±5.1)	71.2(±6.6)	68.9(±3.8)	77.50	77.50	76.25	77.50	77.50
oxxo	97.5	**69.6(±2.0)**	69.5^{1}(±2.1)	63.0^{2}(±3.2)	62.1(±3.8)	56.4(±4.1)	73.75	73.75	71.25	68.75	67.50
xxoo	100	**70.2(±1.5)**	69.3(±1.7)	64.7(±4.0)	68.5(±3.3)	65.0(±4.4)	72.50	72.50	72.50	72.50	72.50
xoox	100	**68.0(±1.2)**	67.8^{1}(±1.2)	63.5^{2}(±3.2)	62.6(±4.2)	59.2(±4.1)	70.00	70.00	70.00	67.50	66.25
oxxx	100	60.9(±0.5)	**61.3(±0.0)**	57.9^{2}(±2.0)	58.5(±3.0)	50.5(±1.7)	61.25	61.25	61.25	61.25	57.50
xoxx	100	59.9(±0.3)	**60.0^{1}(±0.0)**	57.1^{2}(±2.6)	56.8(±3.4)	52.2(±4.2)	60.00	60.00	60.00	60.00	60.00

5 Analysis

In this section we analyse the results of the experiments. In Subsection 5.1 we investigate the relation between the task ambiguity and the performance difference between RL and EL. We delve into the causes of the performance difference in Subsection 5.2.

5.1 Performance Difference and Task Ambiguity

Is there a relation between the proportion of ambiguous states and the performance differences between RL and EL? As shown in Table 1, the absolute difference in mean performance between EL and the RL-methods does not increase with the task ambiguity. However, the absolute difference does not take into account that the maximal possible performance on the task decreases as the task ambiguity increases. Therefore, we measure the 'relative performance difference' r between EL and every other learning method X: $r = (\overline{EL} - \overline{X})/(\max(EL) - 50\%)$, in which $\overline{EL}$ and $\overline{X}$ are the mean performances of EL and method X. We estimate the maximally obtainable performance with the maximal performance of EL, since it is higher than, or equal to, that of the other methods for all sensor configurations. In addition, it corresponds to the maximally obtainable performance for all sensor configurations with two or less functional sensors (as determined by exhaustive search, discussed below). Figure 4 shows the relative performance difference over different levels of task ambiguity. Since there are multiple sensor configurations with a task ambiguity of 95% and 100%, the relative performance differences are averaged for these task ambiguities.

The figure illustrates that the relative difference between EL and the RL-methods increases with the task ambiguity, while the relative difference between EL and RS decreases. Apparently, both EL and RS gain an advantage over RL-methods as the task ambiguity increases. This suggests that direct policy search becomes more important as task ambiguity increases. In the next subsection we analyse why EL can better cope with ambiguity than RL.

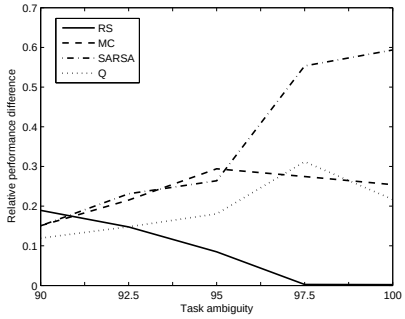

Fig. 4. The relative performance difference over different levels of task ambiguity.

5.2 Causes of the Performance Difference

As mentioned in the introduction, the performance difference between RL and EL is argued to be caused by the fact that RL searches for a value function, while EL searches directly in the policy space. The lower performance of RL might be explained by assuming that for non-Markovian tasks it is more difficult to estimate the rewards associated with state-action pairs. As stated in [1]: *"If ... observations contain very little useful information about the state of the environment, values cannot be reasonably estimated and associated with states, and it may be better to search directly in the space of policies"*. In this subsection, we show that a difficult estimation of the values does not have to be the cause of the lower performance. Instead, we argue that the cause of the lower performance might be that RL's learned policy consists of the state-action pairs with the highest estimated value.

We can verify this for MC by performing exhaustive search on the task and calculating (instead of estimating as MC does) the average rewards of all state-action pairs. Then we can test the policy that combines the state-action pairs with the highest average rewards, and compare it with the maximal performance obtainable on the task. Below, we apply exhaustive search to the sensor configuration 'xoxo'[1]. The four possible sensor states and 21 actions result in a policy space with $21^4 = 194,481$ policies. Table 2 shows the average reward per state-action pair over all policies in the policy space.

The table shows that individual state-action pairs do not have much influence on the reward of a policy. Namely, all average rewards are very close to 50%, while the maximal fitness obtainable on this problem is 77,50%. Apparently, a policy's reward highly depends on the *combination* of state-action pairs that are selected. MC prefers those state-action pairs that have the highest values in Table 2 (shown in bold), which indicates that MC estimates them rather well.

The combination of state-action pairs with the highest estimated values does not result in the optimal policy. Even if MC estimated the average rewards for all

[1] Exhaustive search on the task with other sensor configurations containing one or two functional sensors gave similar results, while exhaustive search on configurations with three or more functional sensors is computationally too expensive to perform.

Table 2. Average rewards for all state-action pairs.

s/a	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
00	0.48	0.47	0.49	0.48	0.50	0.48	0.50	0.49	0.52	0.51	0.52
01	0.49	0.50	0.50	0.49	0.49	0.51	0.51	0.49	0.50	**0.51**	**0.51**
10	0.50	0.50	0.49	0.48	0.49	0.50	0.49	0.48	0.50	0.49	0.49
11	**0.52**	**0.52**	**0.52**	**0.52**	0.51	0.49	0.50	0.50	0.44	0.46	0.47
s/a	1	2	3	4	5	6	7	8	9	10	
00	0.51	**0.53**	0.50	0.50	0.50	0.50	0.48	0.49	0.49	0.48	
01	0.49	0.49	0.50	0.49	0.48	0.49	0.50	0.49	0.48	0.49	
10	0.51	**0.52**	0.49	0.49	0.51	0.50	0.49	0.50	0.49	0.50	
11	0.47	0.43	0.48	0.50	0.50	0.50	0.51	**0.52**	**0.52**	**0.52**	

state-action pairs perfectly, it would not obtain the optimal policy. Namely, the policy that combines all state-action pairs with the highest values only achieves a performance of 51,25% on a whole episode. Hence, the optimal policies contain state-action pairs that do not have the highest values. For example, exhaustive search shows that all optimal policies contain either the action -6 or 6 in the state '00'; actions that do not have the highest values in Table 2. However, the value-distributions for these actions have a higher standard deviation than those for the other actions in this state. The preference of MC for state-action pairs with high values and of EL for state-action pairs included in successful strategies is supported by an analysis of the policies that they learn: for state '00' MC selects actions -2, 0, and 2 significantly more often than EL ($p < 0.05$), while EL selects actions -6 and 6 significantly more often than MC ($p < 0.05$).

The other RL-methods, SARSA(λ) and Q-learning, also learn policies consisting of state-action pairs with the highest values; they only use a different notion of a state-action pair's value. Therefore, the analysis might also be valid for these other RL-methods.

6 Discussion

The results reported in Sect. 4 show that EL performs better than RL on all sensor configurations of the ACP-task. A typical approach for avoiding the problems that arise from perceptual ambiguity is to employ policies that make use of a memory (see, e.g., [1]). Such policies can enhance performance for both RL and EL. For example, recurrent neural networks (policies with a memory) outperform perceptrons (memory-less policies) on the ACP-task [11]: for the configuration without blind sensors this difference is around 2.8%. However, in [6] it is observed that the introduction of a memory does not profoundly reorganise the behaviour of agents performing a certain task, but rather serves to complement the otherwise reactive behaviour. This suggests that behavioural differences between RL and EL may remain when memory is used. Therefore, the choice of learning algorithm might influence the outcome of Artificial Life studies that focus on the type of learned behaviour (e.g., [7]), be it reactive or not.

7 Conclusions

The experimental results reported in Sect. 4 lead to the following two conclusions regarding our research questions. First, RL and EL obtain different performance levels on a non-Markovian task: EL outperforms RL on the ACP-task for each sensor configuration. Second, the relative performance difference between EL and RL increases as the proportion of ambiguous states increases. Our analysis suggests that the increase of performance difference does not have to stem from problems with estimating the state-action values, but from RL learning a policy consisting of the state-action pairs with the highest estimated values. For a non-Markovian task this does not lead to the best performance, since the reward of a policy is highly dependent on the *combination* of state-action pairs. As a consequence, RL and EL exhibit different behavioural strategies on the ACP-task. A possible direction of future research is to determine whether such behavioural differences remain when policies with memory are applied.

References

1. B. Bakker. *The State of Mind: Reinforcement Learning with Recurrent Neural Networks.* PhD thesis, Leiden University, 2004.
2. P. Cohen. *Empirical Methods for Artificial Intelligence.* MIT Press, Cambridge, Massachusetts, 1995.
3. P. A. Crook and G. Hayes. Learning in a state of confusion: Perceptual aliasing in grid world navigation. In *Proceedings of TIMR 2003 - Towards Intelligent Mobile Robots, UWE, Bristol*, 2003.
4. J. Loch and S.P. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pages 323–331, 1998.
5. D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.
6. S. Nolfi. Power and the limits of reactive agents. *Neurocomputing*, 42:119–145, 2002.
7. M. Schlesinger. A lesson from robotics: Modeling infants as autonomous agents. *Adaptive Behavior*, 11:2:97–107, 2003.
8. J. Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In D.S. Lippman, J.E. Moody, and D.S. Touretzky, editors, *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pages 500–506, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.
9. S. Singh, T. Jaakkola, and M. Jordan. Learning without state-estimation in partially observable markovian decision processes. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference (ICML)*, pages 284–292. Morgan Kaufmann, 1994.
10. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, A Bradford Book, 1998.
11. M. F. van Dartel, I. G. Sprinkhuizen-Kuyper, E. O. Postma, and H. J. van den Herik. Reactive agents and perceptual ambiguity. *Adaptive Behavior*, in press.
12. J. Wyatt. Reinforcement learning: a brief overview. In I.O. Stamatescu et al., editor, *Perspectives on Adaptivity and Learning*, pages 243–264. Springer, 2002.