View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by PUblication MAnagement

# A Pilot Study on the Evolution of Reward Signals for Hierarchical Reinforcement Learning

Massimiliano Schembri, Gianluca Baldassarre

Laboratory of Autonomous Robotics and Artificial Life,
Istituto di Scienze e Tecnologie della Cognizione,
Consiglio Nazionale delle Ricerche (LARAL-ISTC-CNR),
Via San Martino della Battaglia 44, 00185 Roma, Italy
{massimiliano.schembri, gianluca.baldassarre}@istc.cnr.it
http://laral.istc.cnr.it/

**Abstract.** Recent research has shown that reinforcement learning agents can by greatly advantaged of the possibility of learning to select macro actions instead, or beside, fine primitive actions. The route usually followed to exploit this idea is to build agents with hierarchical architectures that can learn both a repertoire of macro actions and a macro policy that selects them, on the basis of the "final" reward signals related to the tasks to solve. This research presents a preliminary investigation that follows a different idea: evolving reinforcement signals that allow an agent to learn a repertoire of macro actions in an initial phase of life, and then to assemble and use them to solve different tasks, belonging to the same class, in a succeeding phase of life. The idea is preliminary tested with a simulated robot that has to search targets following coloured trails in a square arena. Results show that the idea is viable and that the emerged reinforcement signals allow the robot to develop macro actions that are actually useful to solve different tasks in succeeding phase of life.

## 1 Introduction[1]

In the last fifteen years research has shown that reinforcement learning systems can greatly benefit of the possibility of selecting macro actions (or "options", or "sub-policies") instead, or aside, actions with a fine granularity ([4] for a review, [14]). In general, macro actions are policies [15], i.e. deterministic or stochastic mappings between environment's states observed by the system and actions that it selects and executes in correspondence to them, which can be selected and executed as wholes by policies of higher level. Usually macro actions tend to last longer than primitive actions, i.e. to involve the selection of more than one primitive action [2] [14].

The advantages of using macro actions instead of (only) primitive actions are several. The first is that they greatly reduce the search space of problems [4] [14]. The second is that they tend to improve the exploration of learning systems as they tend to allow it to "jump" from region to region of the problem space in comparison to primi-

[1] This research has been supported by the Integrated Project "ICEA – Integrating Cognition Emotion and Action", European Commission's grant IP-027819.

tive actions that tend to cause smaller movements in it [8]. The third is that they speed the back-propagation of evaluation signals along the states visited before rewarding states [8].

Macro actions have also drawbacks. One is that they usually require more sophisticated hierarchical architectures with respect to systems using only primitive actions. Another important drawback is that macro actions used by a system needs to be a sub-set of all the possible ones as these are usually very numerous (in fact macro actions are set of state-action couples, and this causes a combinatorial explosion of all the possible combinations). The consequence is that as a system must necessarily use a sub-set of all possible macro actions, it can only find solutions among a sub-set of the possible ones, and this might lead it to find sub-optimal solutions to problems.

The route that the reinforcement learning research usually follows when it uses macro actions consists in building hierarchical systems that learn macro actions and the macro policy to select them at the same time while solving tasks that lead to "final" rewards (e.g., see [2] [4]). Often the goal of this research is to have systems that learn to solve a task faster with respect to plain reinforcement systems using only primitive actions. The general goal pursued here is in part different from this. It consists in designing architectures and algorithms that, given a *class of tasks*, allow a system to find a *repertoire of macro actions* that are *suitable to find near-optimal solutions* to them.

One possible solution to this problem would be using the standard approach mentioned above, based on hierarchical system learning both macro actions and macro policy, to allow a system to solve several tasks of the same class in sequence by preserving the macro actions when passing from one to another with the hope that the system would develop macro actions suitable for all tasks. This approach would be limited by the fact that macro actions developed while solving one task would not be necessarily suitable to solve other tasks, so they would be eventually overwritten and lost (this is an instance of the classic problem of catastrophic forgetting).

Here a different route is followed. A genetic algorithm, which is not affected by catastrophic forgetting, is used to develop macro actions that are suitable for solving few tasks having common features. The genetic algorithm is not used to directly search macro actions of systems, but to find "reinforcers", that is "devices" that produce reinforcement signals that each evolved system uses in its first phase of life ("childhood") to develop macro actions on the basis of a reinforcement learning algorithm. Each system then uses these macro actions in its second phase of life ("adulthood") to solve tasks that deliver "external" reinforcement.

The genetic algorithm is not used to directly evolve the macro actions for two reasons. First, information needed to encode reinforcers in the DNA tends to be much less with respect to information needed to directly encode macro actions. In fact, reinforcers should only encode information that allows the systems to recognize rewarding states, while macro actions should encode sensorimotor mappings usually based on many state-action associations: evolving the former should greatly reduce the search space for the genetic algorithm in comparison to evolving the latter.

The second reason directly involves the ultimate motivation of this work. This work is the first step of a research that aims to develop systems that gather experience by interacting with the environment *without directly tackling specific tasks*. The idea is to mimic humans and other primates that in their first phase of life freely interact

with the environment without attempting to solve specific tasks but in order to acquire skills and knowledge that they will exploit in their adulthood [16]. With this respect, this research is closely related to previous works that presented systems endowed with learning capabilities driven by "curiosity" [7] [10] [12] [13]. In these works systems acquire a *general knowledge* in terms of *capacity to predict* the effects of own actions with the idea that they can exploit this knowledge in a latter stage. More in particular, these systems are endowed with some neural-network "predictors" capable of learning to predict the consequences of the system's actions in a supervised manner (that is of learning the association between current state and action selected for execution on one side, and resulting new state on the other side). The central idea is to train the action selectors of the systems, by reinforcement learning, to select actions that lead them to explore regions of the problem space where the predictors' error decreases quickly (the amount of decrease of the predictors' error is used as a self-generated reinforcement signal). As a result the systems tend to learn to explore regions of the environment where they increase the amount of "knowledge" (i.e. capacity to predict) that they acquire.

This work moves a first step in the direction of building systems that, as the system just reviewed, interact with the environment with the purpose of acquiring general "knowledge" and not to solve directly useful specific tasks, but that, contrary to them, do not learn to predict the consequences of own actions but *learn to act*, in particular *learn repertoires of macro actions* that they might exploit in a later stage to quickly solve new external-reinforcement based tasks. The bet of this research is indeed the hypothesis that the spontaneous interactions with the environment observed in young humans and other primates is mainly directed to this purpose. The reader should keep in mind this wider goals of the work in evaluating the results presented in the succeeding sections, that are little more than a proof of concept of the ideas that this work start to develop.

Section 2 will present the simulated robot and the navigation tasks used to test the system presented here. Section 3 will present the results of the tests while section 4 will close the paper by highlight the strengths and weaknesses of the system.

## 2 Methods

As mentioned, in this work a genetic algorithm is used to evolve the reinforcers of a population of robots, two for each individual. Each robot uses its reinforcers in its childhood to learn two macro actions on the basis of two actor-critic modules. The resulting macro actions can then be used by the robot's "macro actor" (the macro policy) to solve a navigation task during the robot's adulthood. The performance of each robot during its adulthood is then used by the genetic algorithm to select individuals to reproduce. The idea of the whole setup is having a genetic algorithm that selects individuals with *reinforcers* that allow them to develop *macro actions* during their *childhood* that are *good* to solve the *class of problems* they will tackle during their *adulthood*.

**The simulated robot.** The simulated robot is a mobile robot with a 40 cm diameter and a camera assumed to look at a portion of the ground located just in front of the

robot (40×40 cm). At each cycle the robot samples the colour of the ground in 4×4 points distributed on a regular grid overlapping the camera image. Each point can be blue, red or green and there are colour-specific receptors, thus the retina is made of 16x3=48 receptors. Moreover, the robot displaces in space by selecting an orientation change within [-30, +30] degrees, and a step size within [0, +10] cm.

**The task.** In its adulthood the robot navigates in a walled arena that has a ground covered with a texture image of 500×500 pixels (Fig. 1). The image represents two coloured trails, one green and one red, on a blue background. The robot gets a reinforcement signal every time one of the two the front corners of its retina touches a circle target which is randomly positioned on one of the two coloured trail. When the target is reached it is moved to a new randomly selected location. In the first half of the adulthood phase (200,000 cycles), the target's position is chosen within the green trail, while in the second half (other 200,000 cycles) it falls in the red one. Each half of the adulthood life represents a separate learning phase in which the robot has to learn to select the more convenient macro actions from its repertoire. Notice that this task can be considered as a "direction-following" navigation task [6]. The simplicity of the task is due to the fact that it was chosen to run the preliminary tests of the complex architecture illustrated below. Notice that given the set-up, one could expect the robot to develop two kind of macro actions, one consisting in following the red trail and the other consisting in following the green trail, and then that in its adulthood the robot it would have selected the "red following" or "green following" macro actions depending on the position of the targets. This is what actually happened, but with a quite unexpected variation of this basic strategy, as will see in the next sections.

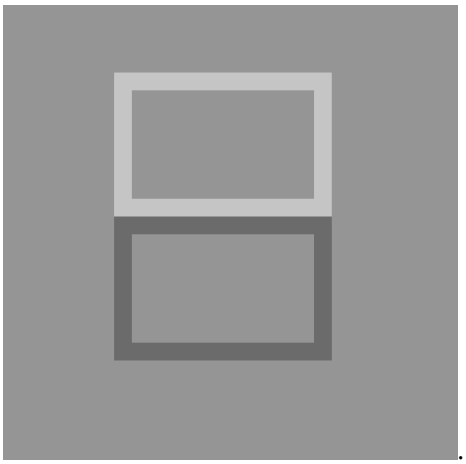
.

Fig. 1. The environment in which the robots were evolved and tested, composed of a walled arena (main grey square), and two circular trails, one green (light grey rectangular path) and one red (dark grey rectangular path).

**The evolved reinforcers.** The mechanism used to generate the reinforcement signals is rather simple. The 48 inputs coming from the retina were weighted and transformed with a sigmoid function into a real number ranging in [0, 1]. The set of weights determined what kind of input the robot preferred and tryed to keep on its retina. This value was multiplied with the robot's speed (normalized in [0, 1]) to have robots moving at maximum speed. The evolution process acted by modifying the weights of each reinforcer which in turn leaded the robot to learn particular macro action.

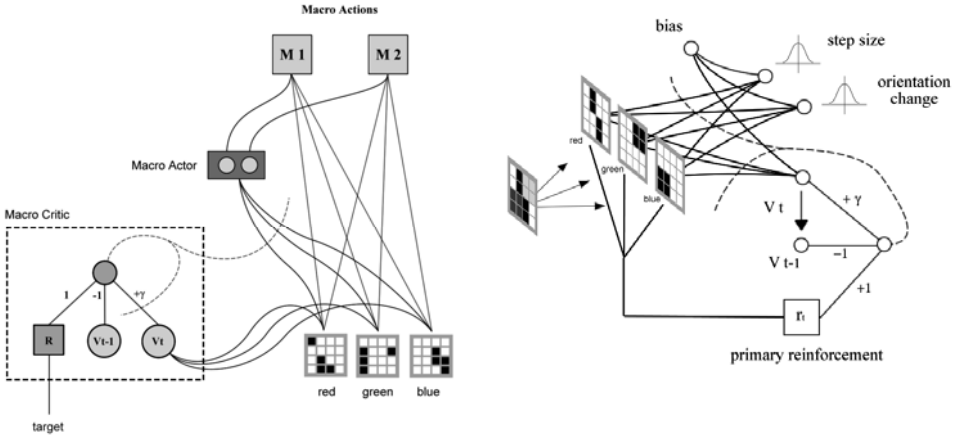


Fig. 1. Left: the whole architecture of the system. Right: the architecture of a macro action.

**The genetic algorithm.** As mentioned, a genetic algorithm is used to evolve the weights of the reinforcers of individuals (two per individual). The genetic algorithm (cf. [99]) evolves a population of 96 individuals for 100 generations. The life of an individual lasted 600,000 cycles and was divided in two phases: (a) childhood phase (200,000 cycles): each of the two macro actions was trained with one of the two reinforces for 100,000 cycles; (b) adulthood phase (400,000 cycles): in this phase the robot learned to select the two already trained macro-actions in order to reach targets. In the first 200,000 cycles the target was randomly positioned on the green trail and in the next 200,000 cycles it was positioned on the red one. The total number of targets reached by the individuals in the adulthood was used by the genetic algorithm to calculate the fitness value and to select the best 16 individuals of the population. Each of those individuals generated 6 offspring. The fitness was computed considering only the target reached in the last 50,000 cycles in each of the last portions of the two adult phase halves, and was normalized [0, 1] on the basis of the maximum theoretical number of targets that the robot could achieve by moving along one of the two trails at maximum speed. The reinforcers of the offspring were obtained from those of the "fathers" by randomly selecting 10% of their weights and by adding a random value in the range [-1.0,+1.0] to them.

**The actor-critic modules that learn the macro actions.** The architecture of the system is illustrated in (Fig. 2). The system has an actor-critic module [55] [15] 15for each reinforcer developed by the genetic algorithm. During the childhood phase the system dedicates 100,000 cycles to train each module on the basis of its corresponding reinforcer (so with two reinforces the childhood lasts 200,000 cycles). Each module that learns a macro action is a neural-network actor-critic model [15]. Each of these modules is composed of two neural networks, an actor and an evaluator. The *actor* is a two-layer neural network with 16 input units, activated by the 16 sampled image point, and two sigmoid output units whose activation is used to decide the robot's orientation change and step size. To this purpose, two random numbers are drawn from two Gaussian function distributions truncated in [0, 1], having standard deviation of 0.3 and centres equal to the activation of the two output units. The *evaluator*, that constitutes the main component of the critic, has the same input units as the actor, and one output unit with Sigmoid transfer function ranging in [0, 10]. The output unit should learn to return evaluations $V_t$ of perceived states. Couples of successive evaluations, together with the reward signal $R_t$ returned by the reinforcer of the macro action, allow computing the critic's surprise $S_t$, a signal used to train both the actor and the evaluator (see below), defined as follows:

$$S_t = (R_t + \gamma V_t) - V_{t-1} \quad \textbf{(1)}$$

where $\gamma$ is a discount factor (set to 0.9 in the simulations).

The actor is trained to select actions that lead the system to rewarding states with a modified error backpropagation rule [11]:

$$w_{jit} = w_{jit-1} + \eta S_t (o'_{jt} - o_{jt})(o_{jt}(1 - o_{jt}))x_{it-1} \quad \textbf{(2)}$$

where $w_{jit}$ is the actor's weights between input unit $i$ and output unit $j$, $\eta$ is a learning coefficient (set to 0.1 in the simulations), $o_{jt}$ it the activation of the output unit, $o'_{jt}$ is the random number drawn from the truncated Gaussian distribution centred on $o_{jt}$, ($o_{jt}$ (1- $o_{jt}$)) is the derivative of the Sigmoid function, and $x_{it-1}$ is the activation of the input unit $i$ at time t-1.

The evaluator's weights are updated with a modified error backpropagation rule [11]:

$$w_{it} = w_{it-1} + \eta S_t (V_{t-1}(1 - V_{t-1}))x_{it-1} \quad \textbf{(3)}$$

where $w_{it}$ is the weight of the evaluator connecting input unit $i$ with its only output unit and ($V_{t-1}$(1-$V_{t-1}$)) is the derivative of the Sigmoid function. This learning rule tends to lead the evaluator to assign evaluations to states that approach the sum of all future discounted rewards (cf. [15]):

$$V_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots \quad \textbf{(4)}$$

Notice that macro actions are trained only during the childhood of the system, not during its adulthood. On the contrary, the trainable macro action is trained each time it is selected for execution by the macro actor. The trainable macro action is implemented through an actor as those of the other macro actions, but it does not posses an evaluator. Indeed, when it is selected by the macro actor at time step *t-1*, it is trained, at the nest time step *t*, on the basis of the surprise of the macro evaluator (see below).

**The macro actor and macro critic components.** These components have the role of learning to select macro actions among those available (including the trainable macro action) at *each step*, and to delegate to them the responsibility to select the actual fine action of the robot. The macro evaluator, the main component of the macro critic, is a neural network with architecture and learning processes identical to those of the evaluators of the macro actions. The macro actor is similar to the actors of the macro actions, but has a Sigmoidal output unit for each macro action (three in this case). The activations of these units are normalized to 1 to generate three probabilities that are then used to randomly draw the actual macro action to which the macro actor delegates the selection of the fine action. The macro actor's weights corresponding to the output unit of the selected action, and only these, are trained as follows:

$$w_{jit} = w_{jit-1} + \eta S_t \left(o_{jt}\left(1 - o_{jt}\right)\right) x_{it-1} \quad (5)$$

where in this case $S_t$ is the surprise of the macro critic. The idea behind this training is that with experience the macro actor learns to pass the control to macro actions that are capable of dealing with certain portions of the environment.

## 3 Results

This section discusses some results of the preliminary experiments run to test the system presented in the previous section. The section first illustrates the dynamics of the evolutionary process and then presents the analysis of the robots' behaviours.

**Analysis of the evolutionary process.** The evolutionary experiment has been replicated 10 times with different seeds of the random number generator. The graph in Fig. 3 shows the fitness of a typical simulation. The average fitness of the population grows quickly and stabilises after the 20$^{th}$ generation. The fitness of the best individual grows quickly as well but keeps a little positive trend until it reaches the maximum value.

Two main results are made apparent from the graph. The first concerns the best fitness: this abundantly overcomes the maximum theoretical fitness value that corresponds to 1.0. This can be explained considering that the maximum fitness was computed assuming a robot that moved at maximum speed exactly *on the trails*. What actually happened was that the evolutionary process found a strategy smarter than this: it found reinforcers, and hence macro-actions, which led the robot to circle inside the trails so as to catch the targets with one of the two front corners of the retina (see below for more details).

The second notable result concerns the average fitness value: this never reaches the level of the best individual. This is due to the high variability in the performance of the robots from trial to trial. To quantify this variability we tested the best individual (i.e. reinforces) of the last generation for 100 trials in the same environment and conditions used during evolution (childhood plus adulthood phases). Fig. 4 shows the fitness relative to these tests in ascending order. The fitness drastically varies from trial to trial ranging from about 0.5 to 1.6. This result reflects the variability of the learning process during the robot's childhood. In fact, even if the reinforcers used by

the robot are exactly the same, different initial conditions at the beginning of the childhood's training period lead to different experiences, and in turn these lead the robot to learn different macro-actions and hence to obtain different fitness levels in the adulthood phase.

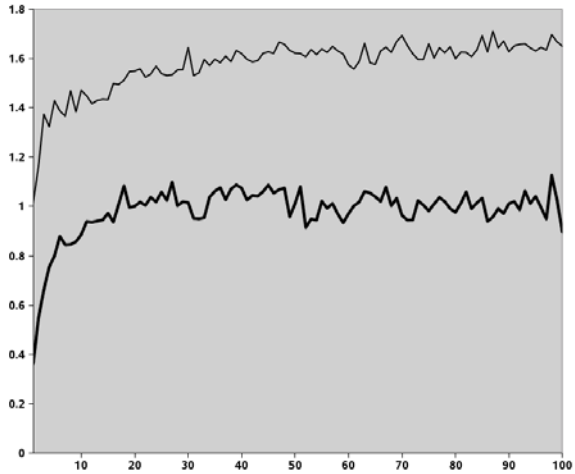


Fig. 3. Graph reporting the best (thin line) and average (bold line) fitness on the y-axix. The fitness was measured during 100 generations of evolution (x-axis).

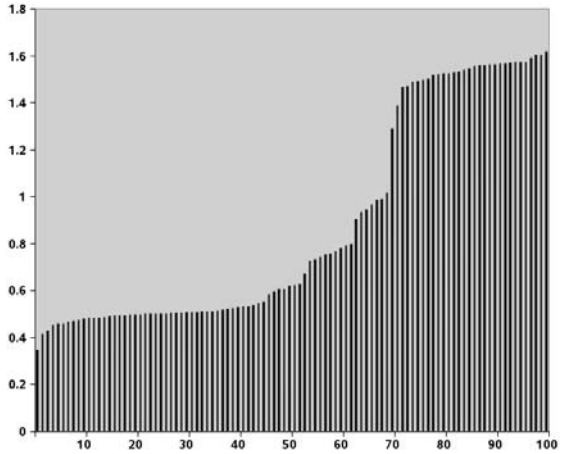


Fig. 4. Fitness of one evolved robot (y-axix) in 100 tests, reported in ascending order.

**Behaviour analysis.** The robot evolves two kinds of reinforcers through which it learns two different macro actions during childhood: one of them consists in following *the edge* of the green trail and the other one consists in following *the edge* of the red trail. The edge is followed by keeping half of the retina activated with a colour (e.g. green) and the other half activated with the background colour, so the robot always circles either clockwise or anticlockwise. Surprisingly, the macro actions are such that not only the robot follows the edges of the trails, but it is also capable of suitably switching behaviour when it happens to follow the edge along the longer, external border of a trails (in particular it switches from following the external edge of the trail to the internal one).

In the adulthood phase the robot learns to select the correct macro action depending on the positioning of the targets: in particular it mainly selects the macro action consisting in following the green trail in the first 200,000 cycles, when the targets are randomly positioned on the green trail, and the macro action consisting in following the red trail in the next 200,000 cycles, when the targets are randomly positioned on the red trail. To analyze the behaviour of the robot, we traced its position at each cycle by drawing a coloured pixel, in correspondence to the centre of its body, indicating which macro-action was selected at this site.

Fig. 5 shows the behaviour of the best individual of the last generation in one of the runs of the experiment. As we are using grey scaled images, we have split the coloured traces into separate images, one for each macro action. The black pixels indicate the places where a particular macro action has been selected and the route followed by the robot. Fig. 5 refers to the beginning of the adulthood training period and clearly shows that the robot starts very early to prefer the first macro action.

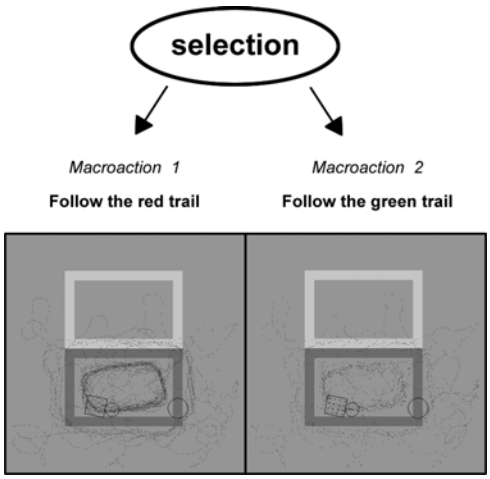


Fig. 5. Macro actions selected by the robot at the beginning of the adulthood training period.

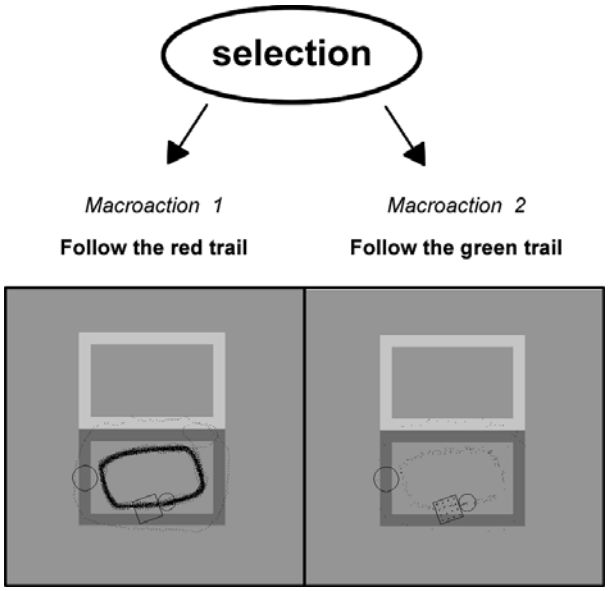

Fig. 6. Macro actions selected by the robot at the end of the adulthood training period.

Fig. 6 shows the last part of the adulthood training period. At this stage the robot clearly selects the macro action which makes it circle inside the red trail. Similarly, in the second half of this phase, when the targets are placed on the green trail, the robot learns to select the second macro action (data not shown).

As mentioned previously, and as shown by these figures, the solution found by the genetic algorithm is rather clever: while trying to touch the target with the front corners of its retina the robot manages to keep its position inside the coloured path, so as to minimise the path and time needed to reach the succeeding targets.

As showed in Fig. 4, the performance of an individual can differ substantially from trial to trial and this is even more evident if one observes the robot behaviour. Fig. 7 shows how the same reinforcer leads to two drastically different macro actions in two different trials. In the first trial the robot learns a useful behaviour that consists in following one of the two coloured trails. In the second trial the robot remains trapped in the border and does not succeed to come back to the centre of the arena. The explanation of this is that the reinforcers lead with a certain probability (about 25%, see Fig. 4) to macro actions that are highly efficient but (e.g., recall the capacity to switch the border of the trail followed) but not very robust, as they lead to bad behaviours the remaining times. A possible explanation of this is that this strategy is "tolerated" by evolution as at least one of the descendants of the same father in one generation have the chance to incur in a suitable childhood's experience, and hence to develop very good macro actions, whereas their "brothers" will receive low fitness and will not reproduce. In this way the behaviours of these lucky, sophisticated individuals will continue to survive along the generations.

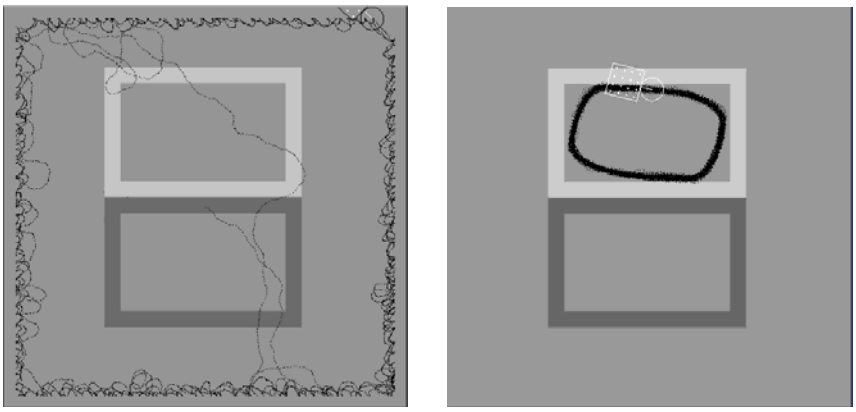

Fig. 7. The same reinforcer produces different macro actions in different trials.

## 4 Conclusions and future work

This paper presented a novel approach and an experimental setup to evolve "reinforcers" that allow a mobile robot to learn macro actions in its first phase of life (childhood), that are suitable to tackle classes of different tasks in its second phase of life (adulthood). Preliminary results suggest that this approach is viable and can be applied to more complex tasks.
Our next objective is to apply this framework to more complex environments in order to try to evolve more than two macro-actions' reinforcers, and in particular to obtain macro actions which are used by the system to solve more than one navigation task.

## References


1. Arbib, M.: Visuomotor Coordination: From Beural Bets to Schema Theory. Cognition and Brain Theory 4 (1981) 23-39
2. Bakker, B., Schmidhuber, J.: Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization. In: Groen, F., Amato, N., Bonarini, A., Yoshida, E., Kröse, B. (eds.): Proceedings of the 8-th Conference on Intelligent Autonomous Systems (IAS-8) (2004) 438-445
3. Baldassarre, G.: A Modular Neural-Network Model of the Basal Ganglia's Role in Learning and Selecting Motor Behaviours. Journal of Cognitive Systems Research 3 (2002) 5-13
4. Barto, A.G., Mahadevan S.: Recent Advances in Hierarchical Reinforcement Learning. Discrete Event Dynamic Systems, 13 (2003) 341-379
5. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike Adaptive Elements that Can Learn Difficult Control Problems. IEEE Transactions on Systems Man and Cybernetics 13 (1983) 835-846

6. Franz, O. M., Mallot, A.: Biomimetic robot navigation. Robotics and Autonomous Systems 30, pp. 133-153, 2000.
7. Kaplan, F. & Oudeyer, P. Iida, F.; Pfeifer, R.; Steels, L. & Kuniyoshi, Y. (ed.) Maximizing learning progress: an internal reward system for development Springer-Verlag, 2004, 259-270
8. McGovern, A., Sutton, R.: Macro Actions in Reinforcement Learning: An Empirical Analysis. Amherst Technical Report Number 98-70. Computer Science Department, University of Massachusetts, Amherst, MA (1998)
9. Nolfi, S., Floreano D..: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press/Bradford Books, Cambridge, MA (2000)
10. Oudeyer, P., Kaplan, F.: Intelligent adaptive curiosity: a source of self-development. In: Berthouze, L., Kozima, H., Prince, C.G., Sandini, G., Stojanov, G., Metta, G., Balkenius, C. (eds.). Lund University Cognitive Studies, Lund (2004) 127-130
11. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Representations by Back-Propagating Errors. Nature 323 (1986) 533-536
12. Schmidhuber J.: Curious Model-Building Control Systems. In Proceedings of the International Joint Conference on Artificial Neural Networks, Vol. 2. IEEE, Singapore (1991) 1458-1463
13. Schmidhuber, J.: A possibility for implementing curiosity and boredom in model-building neural controllers. In: Meyer, J.-A., Wilson, S.W. (eds.): Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats. MIT Press, Boston, MA (1991) 222-227
14. Sutton, R., Precup, D., Singh, S.: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. Artificial Intelligence 112 (1999) 181-211
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge MA (1998)
16. von Hofsten, C.: Eye-hand Coordination in Newborns. Developmental Psychology 18 (1982) 450-461