Automatic synthesis of controllers for real robots based on preprogrammed behaviors

Miguel Duarte, Sancho Oliveira, and Anders Lyhne Christensen

Instituto de Telecomunicações &
Instituto Universitário de Lisboa (ISCTE-IUL)
Lisbon, Portugal
{miguel_duarte,sancho.oliveira,anders.christensen}@iscte.pt

Abstract. We present a novel methodology for the synthesis of behavioral control for real robotic hardware. In our approach, neural controllers decide when different preprogrammed behaviors should be active during task execution. We evaluate our approach in a double T-maze task carried out by an e-puck robot. We compare results obtained in our setup with results obtained in a traditional evolutionary robotics setup where the neural controller has direct control over the robot's actuators. The results show that the combination of preprogrammed and evolved control offers two key benefits over a traditional evolutionary robotics approach: (i) solutions are synthesized faster and achieve a higher performance, and (ii) solutions synthesized in simulation maintain their performance when transferred to real robotic hardware.

1 Introduction

Evolutionary robotics (ER) has been widely researched as a means to synthesize behavioral control for autonomous robots [6]. Artificial evolution has the potential to automate the controller design process without the need for manual and detailed specification of the desired behavior. Artificial neural networks are often used in ER because of their capacity to generalize and to tolerate noise [11] such as that introduced by imperfections in sensors and actuators. Two key issues have prevented evolution from being widely used as an engineering tool for automatic design of behavioral control: (i) bootstrapping the evolutionary process, and (ii) crossing the reality gap, that is, transferring behavioral control from simulation to reality without performance loss. The transition from simulation to real robotic hardware often results in reduced performance or even complete failure due to differences between simulation and the real world [5].

In this study, we propose a novel approach to overcome both bootstrapping issues and to successfully cross the reality gap. We combine artificial evolution with preprogrammed behaviors in order to ensure successful transfer of evolved control from simulation to real robots, while at the same time enable the synthesis of behaviors for relatively complex tasks. We experiment with giving evolution access to sets of simple preprogrammed behaviors, such as *follow wall*, *turn left*,

and *turn right*, which can be switched on and off by a neural network that controls the robot. In this way, we marry the benefits of ER, namely automatic synthesis of behavioral control, with the benefits of preprogrammed behaviors that can be hand-optimized for specific sub-tasks and for the real robotic hardware.

We use the double T-maze navigation task [2] and the e-puck robotic platform [16] for our experiments, because both the task and the robotic platform have been widely studied in the past (see Sect. 2 and Sect. 4 for details). The double T-maze task is a delayed response task in which a robot receives stimuli in the form of light flashes and it must respond by making the correct turns in subsequently encountered T-junctions.

The contributions of this paper are as follows: we propose and study a new approach to the synthesis of behavioral control for autonomous robots. The approach is based on a combination of evolutionary computation and (simple) preprogrammed behaviors. We show that in our approach, solutions are synthesized faster and to a higher quality than solutions evolved using a traditional ER approach, and that the behaviors synthesized in simulation can be successfully transferred to real robotic hardware.

2 Background and Related Work

ER emerged as a field in the beginning of the 1990s [18]. Numerous studies followed which demonstrated robots with evolved control systems solving basic tasks in surprisingly simple and elegant ways. However, to date, only relatively simple tasks have been solved using ER such as obstacle avoidance, gait learning, phototaxis, foraging, and so on [17].

Soon after the research into ER began, two main challenges became clear, namely, (i) that the number of evaluations required meant that simulation had to be used extensively, and (ii) that it often is non-trivial to ensure successful transfer of behavior evolved in simulation to real robots. In [14], three complementary approaches to the evolution of control systems for real robots were proposed: "(a) an accurate model of a particular robot-environment dynamics can be built by sampling the real world through the sensors and the actuators of the robot; (b) the performance gap between the behaviors obtained in simulated and real environments may be significantly reduced by introducing a 'conservative' form of noise; (c) if a decrease in performance is observed when the system is transferred to a real environment, successful and robust results can be obtained by continuing the evolutionary process in the real environment for a few generations."

In 1997, Jakobi [10] advocated the use of minimal simulations to ensure the transferability of controllers evolved in simulation. A minimal simulator only implements the specific features of the real world that the experimenter deems necessary for a robot to complete its task. All other features would either not be implemented or be hidden by an envelope of noise. A number of other approaches to overcome the reality gap include performing evolution directly on

the target hardware instead of in simulation [7], online adaptation through neural plasticity [8], co-evolution of simulators and controllers [3], and promotion of transferable controllers though multi-objective optimization [12]. Conducting evolution on real robotic hardware is, however, tedious and is not always feasible, especially in complex tasks and/or when many trials are needed to reliably estimate the fitness of an individual. In this paper, we propose a novel approach to the engineering of control systems in which we combine simple preprogrammed behaviors with artificially evolved neural networks. Successful transfer of the control synthesized in simulation to real hardware is guaranteed by specifying a set of behavior primitives that have been tested on real robots.

The topic of behavior modularity in the field of robotics has been studied before. Rodney Brooks proposed the subsumption architecture in the 1980's [4]. Brook's approach is characterized by the decomposition of complicated intelligent behavior into many simple behavior modules, which are in turn organized into layers. The layers and ordered in terms of behavioral complexity: the behaviors on the bottom layers are simpler and have a higher priority than the ones on the higher layers. The more complex behaviors can only execute if none of the layers beneath take control of the robot. The concept of behavioral decomposition has also been studied in the field of ER. Moioli et al. [15] used a homeostatic-inspired GasNet with two different behavior controllers (obstacle avoidance and phototaxis) that were inhibited or activated by the production and secretion of virtual hormones. In [13], logic decision trees and task decomposition have been combined with ER techniques. By evolving different sub-behaviors such as "circle box", "push box" and "explore", the authors synthesized a robotic controller that was able to push a box toward a light source.

We use a double T-maze task [2] for our experiments. An example of a double T-maze can be seen in Fig. 1. The T-maze contains three T-junctions. At the start of each experiment, the robot is placed in the "Start zone" and must navigate towards the first junction. On its way, it passes two rows of lights. In each row, one of the lights is activated. The activated light flashes as the robot passes by. The activated light in the first row informs the robot on to which side it must turn in the first T-junction it encounters, while the activated light in the second row informs the robot to which side it must turn in the second T-junction that it encounters. If L1 and R2 are activated, for instance, the robot must make a left turn in the first T-junction and a right turn in the second T-junction so that it reaches exit LR (see Fig. 1), and so on.

Variations of the T-maze task have been used extensively in studies of learning and motivation in animals, neuroscience, and robotics (see [19, 20, 10] for examples). In robotics, T-mazes have been used to study different neural network models such as diffusing gas networks [9], the online learning capability of continuous time recurrent neural networks [2], and the evolution of transferable controllers [10, 12]. However, in the studies where controllers were tested on real hardware, only a single T-maze was used and the mazes were relatively small with respect to the robot. In this study, we synthesize controllers in simulation that enable a real e-puck to solve a relatively large, double T-maze (see Sect. 4).

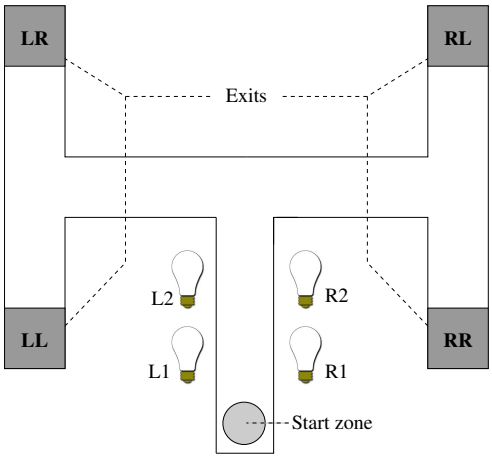

Fig. 1. A double T-maze. A robot is placed in the start zone and must navigate to one of the four exits depending on which lights flash as the robot passes by.

3 Methodology

The main purpose of the proposed methodology is to allow for the synthesis of behavioral control for real robotic hardware. Whereas Jakobi [10] advocated the use of minimal simulations to limit the set of environmental features that a controller can rely on, we suggest limiting the set of actions that a robot can perform. We define a set of behavior primitives based on the robot, its task, and the environment. The behavior primitives are specified in such a way that they have comparable results and performance when executed in simulation and on the real robotic hardware. Conservative noise is added in simulation to promote robustness to the difference that unavoidably exists between the two environments. In this study, the behavior primitives are simple preprogrammed behaviors (*follow wall*, *turn left*, and *turn right*), but they could be more elaborate and even previously synthesized behaviors.

As in a traditional ER setup, the robot's sensory inputs are fed to an artificial neural network. However, instead of controlling the robot's actuators directly, the outputs of the neural network are connected to a "behavior selector" (see Fig. 2). In this study, each output neuron of the neural network corresponds to a single behavior primitive and the primitive which has the highest activation value is executed in a winner-takes-all approach. Some primitives can take more than one control cycle to complete, such as turning 90° left or right. The behavior selector does not execute any other primitive before the previously selected primitive has completed. Alternative behavior selectors could be implemented including selectors that allow for multiple primitives to be executed in parallel. Parallel execution of behavioral primitives could, for instance, allow a robot to communicate at the same time as it executes motor behaviors.

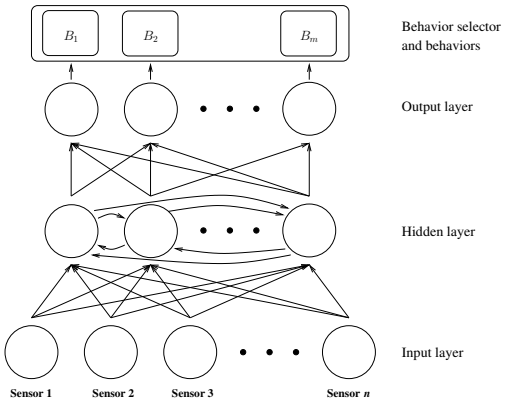

Fig. 2. Example of the controller structure: A continuous-time recurrent neural network [1] receives readings from the robot's sensors. The activation of the neurons in the output layer are fed to the *behavior selector*, which executes one of the behavior primitives based on the activations.

4 Experimental Setup

We used the e-puck [16] for our experiments. The e-puck is a small circular (diameter of 75 mm) mobile robotic platform designed for educational use. For offline synthesis of behavioral control, we use JBotEvolver, an open source, multirobot simulator and neuroevolution framework. The simulator is written in Java and implements 2D differential drive kinematics. Evaluations of controllers can be distributed across multiple computers and different evolutionary runs can be conducted in parallel. The simulator can be downloaded from: `http://sourceforge.net/projects/jbotevolver`.

We built a double T-maze [2] with a size of 2 m $\times$ 2 m (see Fig. 3). In the real maze, the states of the flashing lights are controlled by a Lego Mindstorms NXT brick using 4 ultrasonic sensors and 2 motors (see Fig. 3).

We used four of the e-puck's eight infrared proximity sensors: the two front sensors and the two lateral sensors. We collected sensor samples from two different robots. Each sensor was sampled for 10 samples collected are available at `http://home/iscte-iul.pt/~alcen/sab2012`). At the beginning of every simulation trial, we randomly mapped one of the eight collected sets of samples to each of the robot's proximity sensors. Distance-dependent noise was added to the sensor readings in simulation corresponding to the amount of noise measured during the sampling of the sensors. The light sensor was binary: when a light sensor reading deviated sufficiently from an initial set of readings obtained in ambient light conditions, the robot perceived a flash. In simulation, we added Gaussian noise (5%) to the wheel speed.

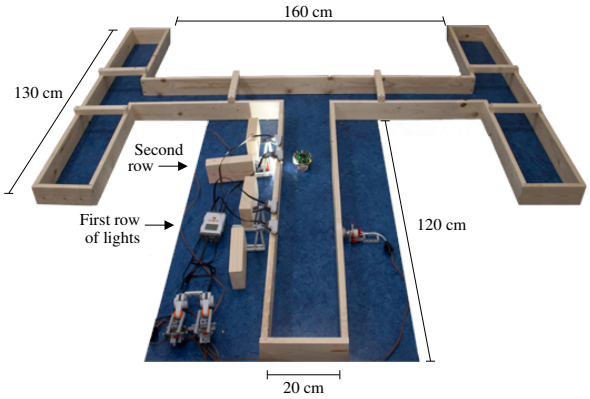

Fig. 3. T-maze with a total size of 2 m × 2 m. The two rows with the lights are located in the central corridor. The first row is at 45 cm and the second row at 83 cm from the start of the maze.

4.1 Controller

The neural controller used in this study is a continuous-time recurrent neural network [1] (see Fig. 2). The input layer of the ANN is composed of 6 neurons: one for each of the four infrared proximity sensors, and one for each of the two light sensors. The readings from the proximity sensors are mapped to distances and then converted to input neuron activations (interval $[0, 1]$). The mapping of readings to distances is done based on the average values for the eight sets of real robot samples. When a light flash is detected, the corresponding input neuron is assigned an activation value of 1.0 for a duration of 15 control cycles.

The hidden layer of the ANN is composed of 10 fully connected neurons. The output layer of the neural network is composed of 3 neurons, one for each of the 3 preprogrammed behaviors available to the network: *turn left*, *turn right*, and *follow wall*. The behavior selector compares the activations of the three output neurons and executes the behavior that corresponds to the neuron with the highest activation. The two *turn* behaviors turn the robot 90°, which takes on average 40 control cycles. During that time, the behavior selector ignores the values of the output neurons in order to allow the turn to complete before executing a new behavior. The *follow wall* behavior moves the robot forward along the closest perceived wall. We limited the speed of the robot to 10 cm/s.

4.2 Evolutionary Algorithm

We train controllers with a simple generational evolutionary algorithm. Each generation is composed of 100 genomes, and each genome corresponds to an ANN with the topology described above. The fitness of a genome is sampled 40 times and the average fitness is computed. Each sample lasts a maximum of 500

control cycles (equivalent to 50 seconds of simulated time). The starting position of the robot is varied up to 5 cm to the left or to the right, and up to 10 cm forward or backward.

The top 5 genomes are selected to populate the next generation using an elitist approach. An offspring is created by applying a Gaussian noise to each gene with a probability of 10%. The 95 mutated offspring and the original 5 genomes constitute the next generation.

The robots are evaluated based on three different outcomes: (i) if they successfully navigate to the correct exit, the assigned fitness is f_1, (ii) if they choose an incorrect exit or colide into a wall, the assigned fitness is f_2, and (iii) if time expires, the assigned fitness is 0. Fitness f_1 and f_2 are defined by:

$$f_1 = 1 + \frac{maxCycles - spentCycles}{maxCycles} \qquad f_2 = \frac{totalDistToExit - distToExit}{3 \cdot totalDistToExit}$$

We ran an additional set of experiments in a traditional ER setup in which the outputs of the neural network controlled the robot's wheels directly. Aside from the difference in the interpretation of the networks output, the experimental setup (network topology, inputs, simulation conditions, and evolutionary parameters) were the same as those described above.

5 Results

We synthesized robotic controllers for a double T-maze task in two different experimental setups: in experimental setup A (Synthesis with Preprogrammed Behaviors), the output of the neural networks activates one of the three possible preprogrammed behaviors, while in experimental setup B (Traditional ER) the output neurons control the wheels of the robot directly.

We conducted 30 evolutionary runs in each of the two experimental setups. Each run lasted 1000 generations. We conducted a post-evaluation of the evolved controllers in which the fitness of every controller was sampled 100 times for each of the 4 possible light configurations. The results are summarized in Figure 4. In experimental setup A, the evolved controllers had an average solve rate of 87%. A solve rate of over 95% was observed in 12 of the 30 controllers. Some of the trials evolved controllers with good solutions as early as the 150th generation.

The solutions produced in different evolutionary runs were similar. The robots learned how to navigate the T-maze correctly, but some of the controllers were not able to use the information from the light flashes to make the correct decisions at the T-junctions, which caused them to navigate to the wrong maze exit.

In experimental setup B, the evolved controllers had an average solve rate of only 42%. The best controller had a solve rate of 88%, and only 12 other controllers were able to correctly solve the T-maze in more than 50% of the samples.

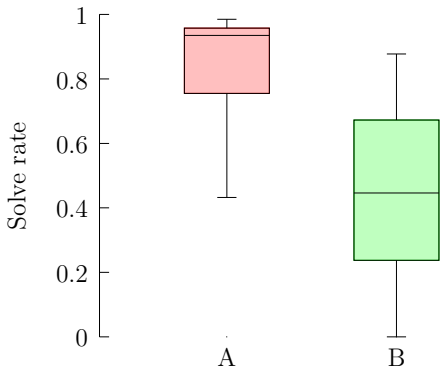

Fig. 4. Summarized results from simulation for setup A and setup B.

5.1 Transfer to Real Robotic Hardware

After the evolutionary process had finished, the 5 highest performing controllers synthesized in setup A, and the 5 highest performing controllers synthesized in setup B were tested on a real e-puck. Each controller was tested 16 times, 4 for each light configuration. The results are listed in Table 1.

All of the 5 controllers synthesized based on preprogrammed behaviors were able to successfully cross the reality gap and solve the real maze consistently. The controllers synthesized in run A22 and A25 managed to solve all 16 samples. The remaining 3 controllers sometimes navigated to an incorrect maze exit: A4 and A13 failed 1 out of 16 samples, and A9 failed 2 out of 16 samples.

The controllers from setup B did not display as high a performance as those synthesized in setup A. Partly, this was because their in simulation performance was not as high as the one in experimental setup A. 4 of the 5 controllers transferred correctly, achieving even comparable performance in reality, but the controller from trial B19 only solved 11 out of 16 samples in the real robot experiments. Videos of the experiments can be seen at `http://home/iscte-iul.pt/~alcen/sab2012`.

Table 1. Summary of the real robot results for the five highest performing evolutionary runs of experimental setup A and experimental setup B.

Evolutionary run	**A22**	**A9**	**A25**	**A13**	**A4**	**Average**
Solve rate (Simulation)	99%	98%	98%	97%	97%	98%
Solve rate (Real robot)	100%	88%	100%	94%	94%	95%
Evolutionary run	**B11**	**B13**	**B19**	**B16**	**B9**	**Average**
Solve rate (Simulation)	88%	86%	79%	70%	70%	79%
Solve rate (Real robot)	100%	100%	56%	75%	75%	81%

6 Conclusions

In this study, we demonstrated how controllers can be synthesized by combining artificial evolution with simple preprogrammed behaviors. Our results show that the proposed approach found good solutions in fewer generations and achieved higher final fitness scores than in a traditional ER setup in which the neural controller has direct control over the robot's actuators. On real robotic hardware, the performance of the controllers synthesized with our approach was similar to their performance in simulation.

We gave neural controllers three simple preprogrammed behaviors: *follow wall*, *turn left*, and *turn right*. If we had used a different set of preprogrammed behaviors, we would potentially have seen different solutions. The solution space is defined by the set of behaviors to which a neural controller has access. This solution space is smaller than the solution space in a traditional ER setup in which the neural controller has direct control over the robot's actuators. The restricted solution space may exclude the optimal solution(s) for a given robot and task. In our study, the controllers that had direct access to the actuators were able to cut corners and continued to move forward while turning in a T-junction. The controllers synthesized in our approach were limited to the *turn left* and *turn right* behaviors that cause the robot to turn 90° on the spot. Consequently, controllers that had direct access to the robot's actuators were sometimes able to complete the task faster than the controllers that were restricted to a predefined set of preprogrammed behaviors.

While the use of a finite set of predefined behaviors may forestall the synthesis of the theoretically optimal controllers, it opens a number of interesting possibilities. Behaviors can be hand-optimized for a particular robot and for particular sub-tasks. For some sub-tasks, it may be relatively easy to rely on artificial evolution to find a good solution, while for others, such as those that are difficult to simulate with sufficient accuracy, may be more easily solved by manually programming a behavior. Moreover, the predefined behaviors need not be limited to simple behavior primitives, but could be controllers that have previously been synthesized by combining other behaviors and so on. In this way, our approach potentially allows for an incremental and a hierarchical synthesis of behavioral control for real robots.

Acknowledgements: This work was partly supported by Fundation of Science and Technology (FCT) under the grants PTDC/EEACRO/104658/2008, SFRH/BD/76438/2011, and PEst-OE/EEI/LA0008/2011.

References

1. R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122, 1992.
2. J. Blynel and D. Floreano. Exploring the t-maze: Evolving learning-like robot behaviors using CTRNNs. In *Applications of Evolutionary Computing*, pages 593–604. Springer, Berlin, Germany, 2003.

3. J. Bongard and H. Lipson. Once more unto the breach: Co-evolving a robot and its simulator. In *Proceedings of 9th International Conference on the Simulation and Synthesis of Living Systems*, pages 57–62. MIT Press, Cambridge, MA, 2004.
4. R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, Mar. 1986.
5. R. A. Brooks. Artificial life and real robots. In *Proceedings of the First European Conference on Artificial Life*, pages 3–10. MIT Press/Bradford Books, Cambridge, MA, 1992.
6. D. Floreano and L. Keller. Evolution of adaptive behaviour in robots by means of Darwinian selection. *PLoS Biology*, 8:1–8, 2010.
7. D. Floreano and F. Mondada. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11(7–8):1461–1478, 1998.
8. D. Floreano and J. Urzelai. Evolution of plastic control networks. *Autonomous Robots*, 11(3):311–317, 2001.
9. P. Husbands. Evolving robot behaviours with diffusing gas networks. In *Proceedigs of the 1sr European Workshop Evolutionary Robotics, EvoRobot98*, pages 71–86. Springer, Berlin, Germany, 1998.
10. N. Jakobi and N. Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6:325–368, 1997.
11. J. Kam-Chuen, C. Giles, and B. Horne. An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on Neural Networks*, 7:1424–1438, 1996.
12. S. Koos, J.-B. Mouret, and S. Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 2012. In press.
13. W.-P. Lee. Evolving complex robot behaviors. *Inf. Sci.*, 121(1-2):1–25, 1999.
14. O. Miglino, H. H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2:417–434, 1996.
15. R. C. Moioli, P. A. Vargas, F. J. V. Zuben, and P. Husbands. Towards the evolution of an artificial homeostatic system. In *IEEE Congress on Evolutionary Computation*, pages 4023–4030, 2008.
16. F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. christophe Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.
17. A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.
18. S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proceedings of the 4th International Workshop on Artificial Life*, pages 190–197. MIT Press, Cambridge, MA, 1994.
19. E. C. Tolman and C. H. Honzik. Introduction and removal of reward, and maze performance in rats. *University of California Publications in Psychology*, 4:257–275, 1930.
20. A. B. L. Torta, M. A. Kramer, C. Thorn, D. J. Gibson, Y. Kubota, A. M. Graybiel, and N. J. Kopell. Dynamic cross-frequency couplings of local field potential oscillations in rat striatum and hippocampus during performance of a t-maze task. *Proceedings of the National Academy of Sciences*, 105(51):20517–20522, 2008.