Artificial Evolution of Robust Adaptive Software: An Application to Autonomous Robots

Dario Floreano and Joseba Urzelai
Center for Bio-inspired Adaptive Machines, Department of Micro-engineering
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland
voice: [+41](21)693-5230; fax: [+41](21)693-3866
e·mail: Dario.Floreano@epfl.ch
www: http://diwww.epfl.ch/lami/team/floreano

Abstract

Artificial evolution of computer software (evolutionary neural networks, genetic programming, evolutionary fuzzy systems, etc.) has been shown to generate software that in many cases is more performant than that designed by engineers. Evolved software performs well under the same conditions used during evolutionary training. However, in situations where unpredictable change may affect normal operation, evolved systems often fail. In this paper we describe a new approach for evolving software that remains adaptive and is therefore very robust to unpredicatble change after evolution. To illustrate the idea, we present the case of evolutionary robots that quickly and reliably adapt online to several types of new situations, including sensory, environmental, and mechanical change while still performing their task. The core of the methodology consists of evolving the mechanisms of parameter adaptation instead of the parameters themselves. We shall conclude by showing how this methodology can be applied to a variety of other situations beyond robotics.

Keywords: Evolutionary Robotics, Evolution and Learning, Adaptive Software, Hebbian Learning, Artificial Life.

1 Automatic Programming

The idea of automatic programming, that is of machines that discover autonomously how to carry out a task, is more than 50 years old, but only in the last few years researchers have successfully implemented it in the form of evolutionary computation [Mit96]. Evolutionary computation is inspired upon the principles of genetic representation and selective reproduction of best individuals. A population of artificial chromosomes, often consisting of a chain of binary digits that encode the properties of the software, is randomly created and the individual software corresponding to each chromosome is tested on a set of tasks. The best chromosomes, that is those software individuals that report the best performance according to a criterion imposed by the user (fitness function), are reproduced by making a number of copies proportional to their performance while other individuals are discarded. Reproduced chromosmes are then allowed to exchange parts among themselves (crossover) and randomly invert the values of some bits (mutation). The new generation of chromosomes is then tested again on the task and the procedure is repeated until a good software individual is found that satisfies the performance criterion.

Artificial evolution has been applied to a variety of systems, such as computer programs, fuzzy systems, neural networks, circuit diagrams, sensory morphology, etc. [NF00] and it has been shown in many cases to generate solutions that are more performant than human designed systems [KKY+00, Kea00]. The limitation of evolved solutions is that they work fine as long as the operating conditions remain identical to those used during evolution. When something changes, the

evolved program might fail to work properly. Unpredictable change is a common problem for machines, but it is even more so for evolutionary machines because evolved solutions capitalize on mechanisms that are not always visible to the user. Therefore, it is much harder to predict under what type of change an evolved solution will fail.

This limitation should be taken into account whenever a machine is expected to cope with sensory data that come from the environment. This is the case, for example, of autonomous robots, computing devices with sensory systems, but also of software for online data mining (such as a database on the web) where input data continuously come in and dynamically change statistical profiles. Considering the rapid spreading of computational devices in everyday use and the fast development of micro sensory devices, the question of how to adapt quickly, autonomously, and online to unpredictable situations becomes a critical issue.

In this paper we present a new approach for evolving software that remains adaptive and is therefore very robust to unpredicatble change after evolution. The core of the methodology consists of evolving the mechanisms of parameter adaptation instead of the parameters themselves. In other words, we evolve the mechanisms by which a machine can online detect the current situtation and modify its own functioning to carry out its original task.

2 Evolution of Adaptive Software

In order to illustrate our methodology, we use a robotic example. Mobile robots are an excellent testbed because their functioning depends on sensors and can move around environments where the conditions and events may unpredictable change. Furthermore, autonomous robots are expected to operate without human intervention and therefore require some level of adaptive intelligence. These constraints are so strong that approaches successful on robots can be extended to a variety of other machines.

In this specific example, the control system is a neural network (in the conclusions of this paper, we suggest how to extend the methodology to other types of adaptive software). Whereas in conventional evolutionary approaches researchers evolve the connection strengths (also known as synaptic values) of the network, here we suggest to evolve simple rules of adaptation that are continuously applied to the synapses while the robot interacts with the environment, *but not the connection strengths*. Whenever an artificial chromosome is decoded into a neural controller, the synaptic strengths are set to small random values. This means that the robot will initially display random actions both at generation 0 and at later generations. While the robot moves, synapses are allowed to change their values every 100 ms (the time necessary for a full sensory-motor loop on the physical robot) using the genetically specified rules. Synaptic change occurs on-line and without external supervision and reinforcement signals during the whole operation of the robot.

We have selected four types of Hebbian rules to be encoded on the artificial chromosome because a) they are strictly local (change depends on the local correlation between the two nodes connected through the synapse); b) they are very simple to implement in software and hardware; c) they are used by almost all biological brains. These rules are plain Hebb rule, Postsynaptic, Presynaptic, and Covariance; interested readers can find detailed information in [FU00]. We have considered different types of genetic encoding, as shown in figure 1. We have either evolved the properties of each inidividual connection (synapse) in the network or only the properties of each neuron. In the latter case, all connections incoming to that neuron will have the same properties and the chromosome is quite short. For what concerns the properties, we have compared three situations: a) Genetically Determined: the chromosome encodes the strengths of the synapses and there is no adaptation (this is the conventional approach); b) Adaptive: the chromosome encodes only the sign, the four learning rules, and four learning rates (this is the core of our new approach); c) Noisy: the chromosome encodes the sign, the strength, and a noise range that can be applied to the synapses at every sensory-motor cycle (this is a control experiment to see whether our method produces results similar to noisy synapses).

A mobile robot Khepera equipped with a vision module is positioned in the rectangular environment shown in figure 2. A light bulb is attached

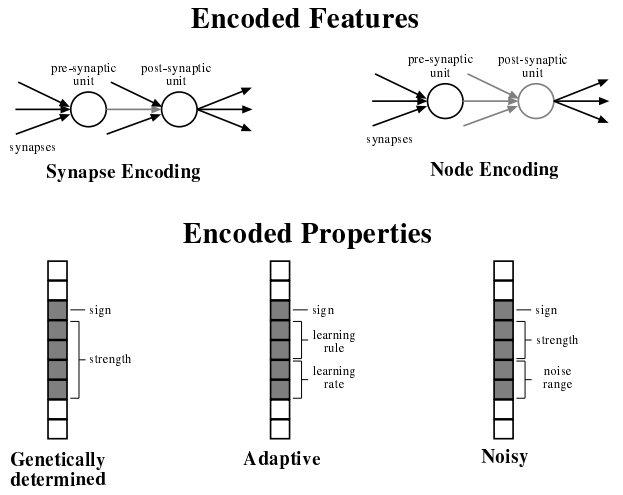

Figure 1: Different type of Genetic Encoding. **Top**: Two feature levels. The genetic string can either encode the properties of each individual synapse in the network (Synapse Encoding) or encode the properties of an entire node and its synapses (Node Encoding). In the latter case, the encoded properties are applied to all incoming synapses to that node. Node Encoding results in shorter genetic strings. **Bottom**: Three types of properties. Genetically-determined properties specify the connection sign and strengths of synapses. Adaptive properties specify the sign, the adaptation rule, and the learning rule of the synapses. Noisy properties specify the sign, weight strength, and a noise range that is continuously applied to the synapse. Properties are applicable to both Synapse and Node Encoding, but in the latter case all incoming synapses will have the same properties.

on one side of the environment. This light is normally off, but it can be switched on when the robot passes over a black-painted area on the opposite side of the environment. A black stripe is painted on the wall over the light-switch area. Each individual of the population is tested on the same robot, one at a time, for 500 sensory motor cycles, each cycle lasting 100 ms. At the beginning of an individual's life, the robot is positioned at a random position and orientation and the light is off.

The fitness function is given by the number of sensory motor cycles spent by the robot on the gray area beneath the light bulb *when the light is on* and divided by the total number of cycles available (500). In order to maximize this fitness function, the robot should find the light-switch area, go there in order to switch the light on, and then move towards the light as soon as possible, and stand on the gray area. Since this sequence of actions takes time (several sensory motor cycles), the fitness of a robot will never be 1.0. Also, a robot that cannot manage to complete the entire sequence will be scored with 0.0 fitness. A light sensor placed under the robot is used to detect the color of the floor—white, gray, or black— and passed to a host computer in order to switch on the light bulb and compute fitness values. The output of this sensor is *not* given as input to the neural controller because we wish that the robot uses only infrared and vision sensors to know its own location. After 500 sensory motor cycles, the light is switched off and the robot is displaced by applying random speeds to the wheels for 5 seconds.

Figure 2: A mobile robot Khepera equipped with a vision module can gain fitness points by staying on the gray area only when the light is on. The light is normally off, but it can be switched on if the robot passes over the black area positioned on the other side of the arena. The robot can detect ambient light and wall color, but not the color of the floor.

The controller is a fully-recurrent discrete-time neural network with 12 neurons that receive information from the active infrared sensors positioned around the robot, a set of ambient light sensors, and a vision module consisting of an array of 64 photoreceptors covering a visual field of 36°. Two motor neurons (M) are used to set the rotation speed of the wheels by mapping the activation of each neuron, normalized between 0 and 1, to a

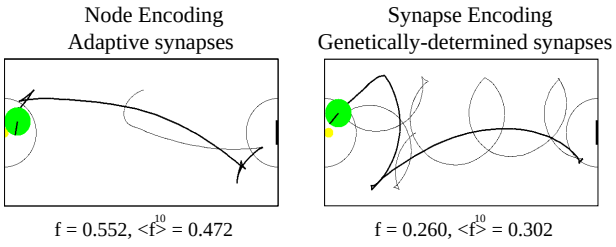

Figure 3: Typical behavior of best individuals of the last generation. **Left**: Node Encoding of adaptive synapses. **Right**: Synapse Encoding of genetically-determined synapses. When the light is turned on, the trajectory line becomes thick. The corresponding fitness value is printed at the bottom of each box along with the average fitness of the same individual tested ten times from different random positions and orientations. The trajectory line is thin when the light is off and becomes thick when the light is turned on.

discrete speed between -20 and 20 (negative values for backward rotation, and positive values for forward rotation). Neurons are synchronously updated every 100 ms during which synapses can be modified according to the evolved learning rules.

Ten sets of experiments have been performed for each of the experimental conditions described in figure 1. The fitness data (not reported here) show that evolution of adaptive synapses using node encoding reports much better and faster solutions than all other conditions. Although conventional evolution of connection strengths can manage to solve the task somehow, it is interesting to see how evolved individuals actually do it.

Figure 3 shows the behaviors of two typical individuals evolved with Node Encoding of adaptive synapses (left) and with Synapse Encoding of genetically-determined synapses (right). Notice that synapses of adaptive individuals are allowed to change during the behavioral tests. The adaptive individual aims at the area with the light switch[1] and, once the light is turned on, it moves towards the light and remains there. Instead, the genetically-determined individual displays always the same looping trajectory around the environment with some attraction towards the stripe and the light (some genetically-determined individuals are not even capable of standing still on the fitness area, result not shown). The *minimalist behavior* of genetically-determined robots, which depends on invariant geometrical relations of the environment, gives them a chance to accomplish the task but with a lower performance. The better fitness of the adaptive controllers (given at the bottom of each box, see figure caption) is given by straight and faster trajectories showing a clear behavioral change between the first phase where the robot goes towards the switching area and the second phase where it becomes attracted by the light.

[1]Its performance is badly affected if the vision input is disabled, indicating that it does not use random search to locate the switch (data not shown).

3 Online Adaptation to Unpredictable Change

Minimalist solutions are often seen as an advantage of evolved systems because they can solve complex tasks by using very simple mechanisms [HHC+97]. Such solutions are often more efficient than human-designed systems because they exploit features of the environment that are not always visible to an external observer. This is also the reason why evolved systems sometimes display innovative and surprising characteristics. However, minimalistic solutions are efficient as lonng as the operating conditions remain the same as those used during evolution. Given the tight coupling between the evolved system and its evolutionary environment, even small change, such as new illumination conditions, can cause a major failure. Furthermore, it is difficult to assess in advance when an evolved system will fail because, as we have just mentioned, an external observer cannot always assess the interdependencies between the evolved system and its environment.

In order to cope with this problem, some authors have suggested to add noise during the evolutionary training [MLN96], to selectively model with variations crucial aspects of the interaction between the evolutionary system and its environment [Jak97], and to continuously test the system in simulated environments that continuously vary [GR92]. Although all these approaches add robustness to evolved solutions, they require the user *to know in advance* what are the crucual aspects of the environment that will change.

Since we are interested in robustness to *unpredictable change*, we have tested evolved controllers under a set of conditions where the envi-

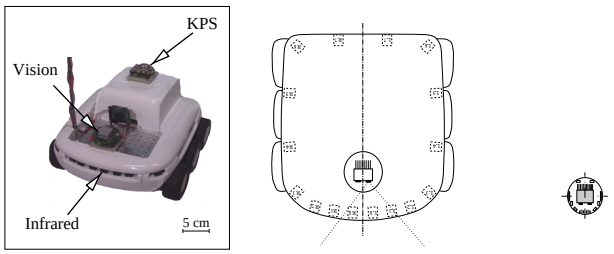

Figure 4: The Koala robot used in the cross-platform experiments. **(a)**: The controller receives the activation of the infrared sensors (with different detection range and response profile to those used by the Khepera) and the linear vision camera (same of Khepera) as input and generates motor commands for the robot. The localization module (KPS) provides the position of the robot at every time step in order to plot its trajectory. **(b)**: Sensory configuration of the Koala robot. **(c)**: Size of the Khepera robot compared to the Koala robot.

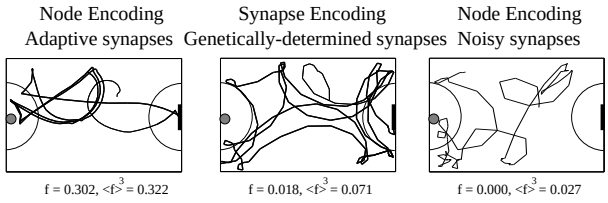

Figure 5: Behavior of best individual of the last generation evolved in simulation for the Khepera robot, and tested on the physical Koala robot. **Left**: Node Encoding of adaptive synapses. **Center**: Synapse Encoding of genetically-determined synapses. **Right**: Node Encoding of noisy synapses. The trajectory line is thin when the light is off and becomes thick when the light is turned on. The corresponding fitness value is printed at the bottom of each box along with the average fitness of the same individual tested three times from different random positions and orientations.

ronment and the robotic platform were changed in ways that had not been included during evolutionary training. More specifically, we compared best evolved controllers obtained with a conventional approach (genetically-determined) with best controllers obtained by evolving the adaptive mechanisms of the controller. The robots have been tested in the following conditions: a) new sensory appearances (changing the color of walls); b) transfer from simulations to real robots (evolving in simplistic simulations and then transferring evolved controllers to physical robots); c) cross-platform transfer (transferring an evolved controller on a different robotic platform); d) new spatial relationships (changing the position of the light switcher and of the light bulb). In all cases, evolved adaptive controllers succeeded to perform the task in the new environments by autonomously modifying their connection strengths on-line without requiring additional evolutionary training whereas genetically-determined individuals failed most of the time or reported much lower performance.

Here we describe only the results on cross-platform transfer. The best controllers evolved on the Khepera robot were transferred on the Koala robot which is much larger, has a different morphology, a different sensory layout, and a different sensor response profile (figure 4). The Koala robot was positioned in a large office-size arena and its positioned recorded every 100 ms while each of the two evolved best controllers attempted to perform the task.

Adaptive individuals correctly approach the light-switching area and then become attracted by light (figure 5, left). Since the response profile of the motors is mcuh different than that of the Khepera robot, once arrived under the light the evolved adaptive robot moves around the fitness area while remaining close to it until the testing time is over. This is the only reason why their performance is slightly inferior to that measured during evolution on the Khepera. On the other hand, genetically-determined individuals (center) perform looping trajectories around the environment and do not display any attraction by the black stripe or the light. They eventually manage to pass through the light-switching area, turn the light on, and occasionally score some fitness points passing over the fitness area by chance. In several cases, genetically-determined individuals get stuck against the walls of the environment (behaviors not shown). Individuals with noisy synapses (right) score low fitness because their strategy consists of random movements.

4 Conclusion

We have described a method to evolve the mechanisms of online adaptation for an autonomous robot and showed that this methodology generates better solutions that can adapt to unpredictable change without requiring additional time-consuming evolution. We believe that this approach can be extended to a variety of other computational mechanisms. The most important thing is to evolve local rules that can modify the system parameters according to the correlation between interdependent elements.

For example, one may apply this method to evolutionary electronic circuits, such as FPGAs. Instead of evolving the connectivity of the circuit logic cells, one may encode rules for fast rewiring depending on the correlated activations of linked cells. Technically speaking, this is feasible because nowadays FPGAs can be entirely reconfigured in less than 1 millisecond. By evolving mechanisms of on-the-fly rewiring, instead of fixed configurations, evolved FPGAs may display the same type of adaptivity than our robots described above.

Another possibility is to evolve computer programs where functions that depend upon several variables can adapt online their output depending on the correlated values of these variables as they come in. This would be a quite simple extension to Genetic Programming that currently evolves only the functions themselves, but not their internal adaptation mechanisms. An application of this approach would be online data-mining for the web where input data come in at different rates and continuously change the probability distribution function.of the database and are therefore hard to analyze with pre-defined programs and conventional machine learning. Yet another application is computer programs that must interact with user-driven inputs, adapt to them and still carry out their predefined task.

Acknowledgments

Dario Floreano acknowledges support from the Swiss National Science Foundation, grant nr. 620-58049. Joseba Urzelai is supported by grant nr. BF197.136-AK from the Basque government. We thank Patrick Saucy for his help on the cross-platform experiments.

References

[FU00] D. Floreano and J. Urzelai, *Evolutionary Robots with Online Self-Organization and Behavioral Fitness*, Neural Networks **In press** (2000).

[GR92] J. J. Grefenstette and C. L. Ramsey, *An approach to anytime learning*, Proceedings of the Ninth International Machine Learning Conference (ML'92) (San Mateo, CA) (D. H. Sleeman and P. Edwards, eds.), Morgan Kaufmann, 1992, pp. 189–195.

[HHC+97] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi, *Evolutionary Robotics: The Sussex Approach*, Robotics and Autonomous Systems **20** (1997), 205–224.

[Jak97] N. Jakobi, *Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics*, Proceedings of the 4th European Conference on Artificial Life (Cambridge, MA) (P. Husbands and I. Harvey, eds.), MIT Press, 1997.

[Kea00] O. Keane, *Thinking machines*, Business Week **August 7** (2000).

[KKY+00] J. R. Koza, M. A. Kean, J. Yu, F. H. Bennett, and W. Mydlowec, *Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming*, Genetic Programming and Evolvable Machines **1** (2000), 121–164.

[Mit96] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[MLN96] O. Miglino, H. H. Lund, and S. Nolfi, *Evolving Mobile Robots in Simulated and Real Environments*, Artificial Life **2** (1996), 417–434.

[NF00] S. Nolfi and D. Floreano, *Evolutionary robotics: Biology, intelligence, and technology of self-organizing machines*, MIT Press, Cambridge, MA, 2000.