Berthouze, L., Kaplan, F., Kozima, H., Yano, H., Konczak, J., Metta, G., Nadel, J., Sandini, G., Stojanov, G. and Balkenius, C. (Eds.)
Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems
Lund University Cognitive Studies, 123. ISBN 91-974741-4-2

Speeding up Learning with Dynamic Environment Shaping in Evolutionary Robotics

Nicolas Bredeche
Inference and Learning group - TAO/INRIA
LRI - University Paris-Sud
bredeche@lri.fr

Louis Hugues
Laboratoire d'Informatique de Paris 6
louis.hugues@wanadoo.fr

Abstract

Evolutionary Robotics is a promising approach to automatically build efficient controllers using stochastic optimization techniques. However, works in this area are often confronted to complex environments where even simple tasks cannot be achieved. In the scope of this paper, we propose an approach based on explicit problem decomposition and dynamic environment shaping to ease the learning task.

1. Step-by-step learning task

Evolutionary Robotics offers an efficient and easy-to-use framework for automatically building controllers for an autonomous robots. However, a major drawback of this approach relies in the difficulty to define the fitness function (i.e. the learning setup) in order to get satisfying results. As a consequence, many works have been limited to simple problems such as wall-avoidance or target-following (Nolfi and Floreano, 2000).

In the scope of this paper, we are interested in the decomposition of a straightforward learning task within a complex environment into learning subtasks that can be easier to achieve from the robot viewpoint. A typical task we address here is that of a robot that learns to explore a complex environment so as to find food items as well as learning other behaviors. The proposed approach is based on incremental learning and dynamic environment shaping so as to simplify both learning tasks and environmental conditions in order to speed up learning. In order to do so, we first introduce the concept of parameter optimization for an Environment Generator (i.e. a function that instantiates an environment from a class of target environments). Then, we suggest to define learning stages to build increasingly complex controllers thanks to task decomposition and environment shaping in order to fit the problem to controllers at hand.

Our approach relies on the use of a dynamic fitness that acts both on the reward given to the robot as well as on the very environment used to train the robot. The idea is that the environment has to provide paths to the design of an efficient robot controller. The primary objectives are to design an environment generator for the task at hand as well as to identify optimization parameters for this generator that may help learning from the agents point of view, even in the worst case (i.e. speeding up random search by ideally limiting the environment to the very specific sub-problems to solve). In order to do so, we define the following :

- The **Environment Generator** is a function that instantiates environments from a class of environments (i.e. maze environments, human-like environments, extreme environments, etc.). Parameters that control the environment generator are tuned during evolution so as to limit environment/task complexity so as to be the most suited to the current generation with regards to the learning task;

- a set of **sub-fitnesses** that take part in the global dynamic fitness and *weighted sub-fitnesses combination patterns* that evolves through time according to statistical information on the population performance. Accordingly we define the following :

  dynamic fitness : *balanced combination of fitnesses* : basic tasks should be the completed first (e.g. *wall-avoidance*). This dynamic fitness is tuned during the course of optimization by switching from one learning stages to another as defined thereafter;

  learning stages : *online balance of fitnesses weight that changes according to a given order.* The goal is to speed up learning thanks to a collection of fitnesses for which the supervisor defines automatic activation patterns depending on the agents performance at hand (e.g. *avoid-wall is the first and only fitness as long as relevant avoidance behavior has not been found - this favor small environments and running time parameters for example*).

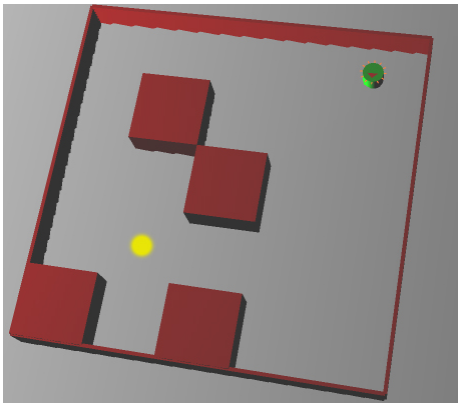

Figure 1: The task : a robot has to find and eat the yellow item in a maze environment.

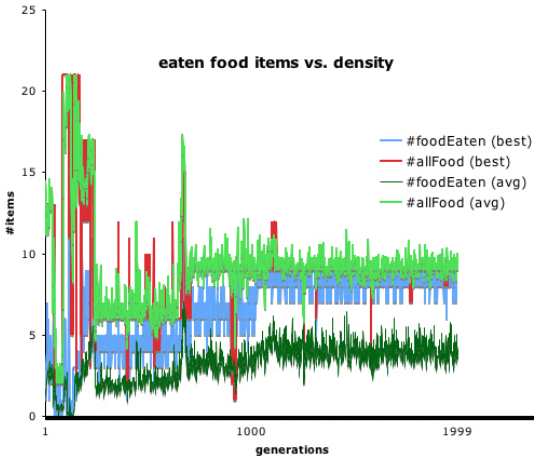

Figure 2: Exp. 1 : eaten food items and food density. There can be up to 21 food items in the environment – the optimal setup occurs when the environment contains only one food item that is eaten by the robot for any combination of robot starting and food positions. In this case, agent self-tuning the paramaters of the environment and task difficulty leads to compromise and local minima (i.e. learning stages must be fixed externaly).

As a consequence, agent population shall alternate between competition to optimize behaviors at a given stage of development (i.e. a specific set of parameters for the agent and the environment) and competition to explore new behaviors afterwards for which rewards is less important but indeed favor exploratory agents that succeeds over the rest of the population (i.e. selection pressure forces to go further even for little gain while preserving what has been learned). It is possible to address both *concurrent* and *sequential tasks* by tailoring the environment and tuning the fitness so as to temporaly define a sub-task that focus on the very sub-problem at hand.

2. Preliminary Experiments

To illustrate our approach, we have set up the task illustrated in fig. 1, that is a robot looking for food in a maze-like environment. The goal is to get the food; the food can only be seen when the robot is on it; the robot has to perform a four steps sequence to eat the food; food location is randomly chosen every run; the target *eat sequence* is randomly chosen before starting evolution.

The robot controller is divided in two sub-controllers. Firstly, a classic three layers feed-forward neural networks with four input nodes encoding the four infra-red quadrants of the robot, four hidden nodes and two output motor nodes, one for translational velocity and the other for rotational velocity (like a Khepera 2 mobile robot). Secondly, the *eat sequence* is encoded into the robot genome and is automatically performed whenever the robot is near some food. The robot plays the whole sequence to the end before it can (or cannot) eat the food at hand. Each part of a sequence is a specific hard-coded behavior chosen among a library of 16 possible behaviors (e.g. *360 degree turn to the right, 90 degree to the left, etc.*). As a consequence, the food is always removed (either lost or eaten).

Preliminary experiments showed the differences between a straight-forward fitness, a dynamic fitness where each individual may tune the environment and a dynamic fitness which is tuned along with the environment by an automatic supervisor (i.e. fitness and environment parameter tuning is performed according to predefined performance statistics based conditions). Optimized parameters include both neural networks weights, eat sequence parameters and environment parameters (food density, epoch duration and eat sequence complexity). These preliminary experiments have shown that :

- fitness decomposition leads to faster learning thanks to simpler problem upon which optimization process may concentrate (in this case, this is due to the fact we rely on population-based stochastic optimization techniques, i.e. simpler problem avoid sparse population);
- leaving up to each individual the choice of tuning the environment parameters leads to compromise and local minima in the course of optimization (see fig. 2). As a consequence, it is better to outsource the tuning of the task difficulty to an external process, either through a predefined dynamic fitness or co-evolved tasks vs. agents setup.

References

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics, The Biology, Ingelligence, and Technologie of Self-Organizing Machines*. MIT Press.