Evorobot*
A Tool for Running Experiments on the Evolution of Communication

Stefano Nolfi and Onofrio Gigliotta

Institute of Cognitive Sciences and Technologies, CNR

1 Introduction

This document introduces the Evorobot* software, a tool developed at the Laboratory of Artificial Life and Robotics, CNR-ISTC (`http://laral.istc.cnr.it`) by Stefano Nolfi and Onofrio Gigliotta, that will allow you to run experiments on the evolution of collective behavior and communication (for more information about evolutionary robotics see Nolfi and Floreano, 2000). The tool is based on the e-puck robotic platform developed at the Ecole Politechnique Federale de Lausanne (see `http://www.e-puck.org/`). The tool allows to evolve neural controllers for this type of robotic platform both in simulation and in hardware.

The Evorobot* software is copyrighted (or ”copylefted”, to use the term introduced by the Free Software Foundation) under a GNU General Public License. This means that it may be used, copied, modified, or redistributed for free. However, any redistribution (of the original or modified code) should adhere to the General Public Licence terms, and copies should acknowledge the original authors and be subject to the terms of the GNU General Public License. The Evorobot* package (which include the source files, the user manual, a set of examples, and a tutorial) can be freely downloaded from `http://laral.istc.cnr.it/evorobotstar/`.

The Evorobot* software is written in C and C++ and based on the QT graphic tools. The software can be compiled under Windows, Linux, and Mac operating systems. The instructions on how to compile the program and an explanation of the content of the source file can be found in the user manual of the program.

This document provides a brief overview of the program and a tutorial which will allow you to familiarize with the tools and to understand how you can run your own experiments.

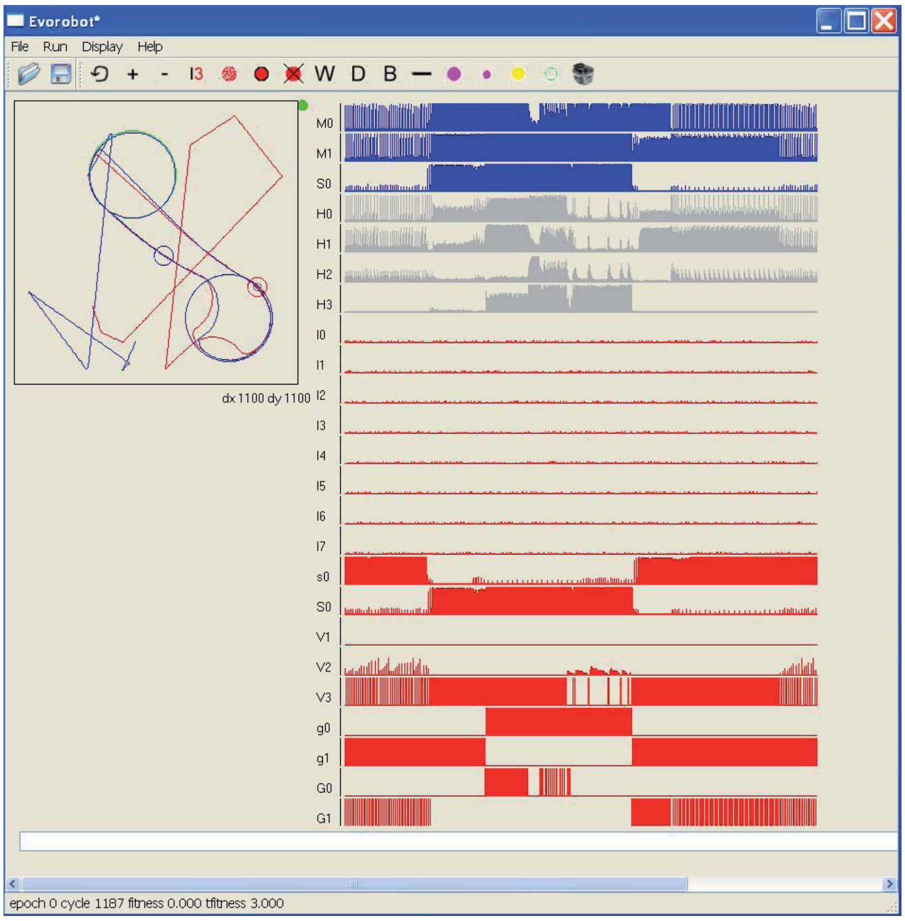
2 Evorobot* features

Evorobot* allows you to evolve a robot or of a group of robots for the ability accomplish a certain task in a given environment. The robots have a circular body shape (i.e. corresponding to the characteristics of the e-puck robotic platform) and can have different types of sensors (e.g. infrared, ambient light, ground, vision, signal) and of actuators (e.g. wheel motors, active led lights, signal emitters). The robots are provided with neural controllers including a certain number of sensory neurons (which encode the state of the corresponding sensors), internal neurons, and motor neurons (which encode the state of the corresponding actuators). The environment consists of one or more arenas surrounded by walls which can include objects (e.g. walls or cylinders with different size and color), light bulbs, and landmarks (e.g. floor areas painted in a given color). The task to be accomplished is specified in a fitness function which determines how the performance of the evolving robots will be evaluated.

More precisely, the Evorobot* software include six integrated tools (i.e. program sub-parts playing specific functionalities): (i) an evolutionary algorithm, (ii) a neural network simulator, (iii) a simulator of the robots, of the environment, and of their interaction, (iv) a graphic interface as well as commands for saving and analyzing data, (v) a tool that allows the user to test and/or evolve robots' controller in simulation and in hardware, (vi) an evorobot* firmware to be loaded on each robot that allows each robot to behave on the basis of the neural controllers evolved on a PC which communicate with the robots through a wireless bluetooth connection.

The evolutionary algorithm. The evolutionary algorithm tool allows the user to create an initial generation of genotypes, to evaluate individuals' performance on the basis of a fitness function, and to generate successive generations. Each selected individual is allowed to produce a certain number of offspring which consists of copies of the genotype of the reproducing individual with the addition of variations (i.e. mutations). The user can specify the parameters of the evolutionary process including the number or reproducing individuals, the number of offspring, the mutation rate, the use of elitism, etc. The user can also specify the number of individual robots which are situated in environment and whether the group of robots is homogeneous or not (i.e. whether the individuals have the same genetic characteristics or not).

The neural network simulator. The neural network simulator tool allows the user to specify the characteristics of the robots neural controller (i.e. the architecture of the neural controller and the number and type of the neurons) and to compute the activation state of the neurons. The program allows the usage of standard logistic neurons, leaky neurons with genetically encoded time constant parameters, and biased or unbiased neurons. Moreover, the program allows the users to easily specify any possible type of neural architecture. Standard architectures (e.g. feed-forward or recurrent neural controller) can be specified by setting few parameters. Irregular or unconventional architectures can be specified by indicating connectivity blocks formed by a group of neurons receiving connections from another group of neurons. The tool includes a graphic interface which allows the user to easily define the architecture of the robots' neural controller as well as display the architecture and the parameters of a specific individual controller.



epoch 0 cycle 1187 fitness 0.000 tfitness 3.000

Fig. 1. Evorobot* is a standard graphic application with a menu bar, a tool bar, and a status bar. The Figure shows the typical information which is displayed when the user test a group of robots in simulation. The top-left side of the graphic window displays the environment and the robots' behavior in the environment. The right side displays the state of the neurons of the robots' neural controllers while the robots move in the environment. The status bar at the bottom of the picture shows the current trial, cycle, and fitness values.

The robot/environmental simulator. The robot/environmental simulator tool allows the user to define the characteristics of the robots and of the environment (i.e. the robots' sensors and actuators, the size and the objects contained in the environment), to compute (in simulation) the state of the robots' sensors (on the basis of the current position and orientation of the robots in the environment) and how the position and the orientation of the robots or the characteristics of the environment vary in simulation as a result of the robots' actions.

The graphic interface. The graphic interface (see figure 1) tool allows the user to: (i) run commands from the menu bar, (ii) modify the parameters of the program while the program is running by using a simple command line instructions, (iii) modify the characteristics of the environment, the positions of the robots, or the characteristics of the neural controllers graphically, (iv) visualize graphically the environment, the robots' behavior, the architecture of the neural controller, the current state of the neurons, the fitness value, etc.

The Evorobot firmware.* The evorobot* firmware is a software which allows to run evolutionary experiments on the robots or to test neural controller evolved in simulation on the real robots. To use the real robots, the user should load the firmware on each robot and establish a bluetooth connections with the robots. A part from that, the user can use the same graphic interface and the same command for running experiments in simulation or in hardware.

The six tools described above are tightly integrated to maximize usability and to reduce the risk of introducing errors while extending or modifying the source code. In particular, the program automatically determines the length of the individuals' genotype on the basis of the characteristics of the robots' neural controllers. Moreover, the program automatically creates sensory and motor neurons on the basis of the sensors and motors selected by the user. Finally, the graphic interface of the program automatically displays all crucial variables over time thus allowing to the user to quickly analyze the obtained results and eventually easily identify problems or bugs introduced in the code.

3 Using Evorobot*

The use of the tool typically involves three phases: (i) setting-up an experiment, (ii) running the evolutionary process, and (iii) analyzing the obtained results. The program allows the user to replicate some of the experiments described in this book, but also to run its own new experiments. In some case, new experiments can be set up simply by varying the program parameters. In other case (e.g. when the user wants to use a new fitness function) the user might need to extend the source code of the program and recompiling it before running the new experiment.

Setting up an experiment. During this phase the user define the characteristics of the robots and of the environment which are fixed by specifying the sensory-motor system of the robot, the architecture of the neural controller and the type of neurons, the characteristic of the environment/s, the characteristics of the robots' lifetime (e.g. number or robots concurrently situated in the environment, number of trials, number of step for each trial, fitness function type), and the characteristics of the evolutionary process (e.g. population size, mutation rate, number of generations, etc.). These parameters can be defined by editing the configuration text files before executing the program or by modifying the parameters through the graphic interface and by saving the modified parameters in the configuration files. In the latter case, once the parameters have been set, the users should quit from the program, and re-execute it again in order to allow the program to appropriately allocate memory on the basis of the parameters which have been specified. The parameters which do not affect the size of the genome or the size of the architecture of the neural controller (i.e. the one that do not affect memory allocation) instead, can be modified at any time without the need to quit and restart the program.

Running an experiment. To run an evolutionary experiment the user simply has to issue the corresponding command from the menu bar. During an evolutionary process the program automatically display statistical data about the fitness and automatically save the genome of evolving individuals.

Analyzing obtained results. The program allows the user to test the behavior of a single or of a group of robots in simulation and in hardware and to run evolutionary experiments (typically in simulation). To test pre-evolved individual the users should first load the corresponding genotype from a file. The program also allows the user to easily analyze evolved robots at the level of the robots' behavior but also at the level of the robots' neural controller. In particular the program allows the user to analyze the free parameters of evolved neural controllers, to lesion neurons, to visualize the state of the neurons while the robots interact with their environment etc.

Extending the Evorobot source code.* The user manual of the program provides an overview of the program source code as well as indications for the most common extensions required which typically consist in the need to implement a new fitness function, or new type of sensor or actuator. Evorobot* has been conceived in order to simplify these operation as much as possible.

4 User Manual, Tutorials & Download Instructions

Evorobot* can be downloaded from `http://laral.istc.cnr.it/evorobotstar/`. The package includes the program, the source files, a series of pre-prepared examples which allow you to replicate some experiments published in scientific journals, a user manual (providing a detailed description of all functionalities, commands, and variables), and a tutorial that will allow you to familiarize with the tool by running and analyzing some experiments on the evolution of communication.

Bibliography

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics. The biology, intelligence, and technology of self-organizing machines.* MIT Press, Cambridge, MA.