# Evolving a heterogeneous foraging robot task

Fernando M. Montes González, Fernando Aldana Franco, Luis Felipe Marín Urías, and Homero Vladimir Ríos Figueroa.

Facultad de Física e Inteligencia Artificial, Universidad Veracruzana
Sebastián Camacho No. 5, Centro, Xalapa, Ver., México
fmontes@uv.mx, fernando_aldana_franco@hotmail.com
*(Paper received on August 30, 2010, accepted on October 20, 2010)*

**Abstract.** In this paper we develop an environmental setup for experimenting with both the e-puck and the Khepera robot platforms. We have used the evolutionary robotics approach to develop a foraging task where the robot approaches food and avoids poisonous food. The use of the EvoRobot* simulator facilitates the implementation of lights as poisonous food and cylinders as healthy food. Later on, the robustness of the Neural Network is tested using the evolved controller in the Khepera robot.

**Keywords:** Mobile Robots, Evolutionary Robotics, Genetic Algorithms.

## 1 Introduction

The Evolutionary Robotics (ER) approach commonly relies on the implementation of an evolvable robot controller which produces the fittest individuals after several simulator interactions. In general an average population to be evaluated forms the elementary building blocks [1] that will serve as the necessary leverage to increment the overall fitness population. Parameter interactions in the implementation of a robot controller epistatically increment the chances to find near-optimal solutions. Most of the times a single objective evaluation function is employed in ER due to the fact that the resultant robot behavior comes from a dynamic system made with the robot and its environment [2]. However, in evolutionary computation is possible to simultaneously optimize several objectives without aggregating them as a single monolithic fitness function [3]. Recent research relies on the concept of domination and generates the so-called Pareto front to bootstrapping a light-seeking robot behavior [4]. Here we use a single objective function to reward the performance of global behavior; thus we start by designing a foraging robot task. Section 2 introduces some necessary background on genetic algorithms for the development of the experiments below described. In Section 3 we present the evorobot simulator used for our experiments. Then, in section 4 we explain the implementation of the foraging task for the e-puck and the Khepera robots. Next, section 5 presents the results of setting up a robot in a foraging task with non-homogenous food. Finally, in section 6 we provide a general discussion highlighting the importance of our work.

## 2 The use of Evolution in Robotics

The use of evolutionary techniques for the development of robot control systems commonly relies on the use of neural networks [5]. Therefore, a population of robot controllers is encoded into phenotypic representations, and then three operators known as reproduction, crossover and mutation, are applied to the population in order to produce a new offspring. The decoding of every genotype into individual phenotypes aids the evaluation of a new offspring by allowing each individual to live for a limited period of time. This process is repeated until a satisfactory evaluation level is obtained. In robotics we can shape behavior with high-quality results using a gradient descent computational method known as Genetic Algorithms (GAs) [6]. In order to find a solution within a search space a set of biologically inspired operators are applied to move across a convoluted landscape. This surface is the result of measuring over an evolutionary timescale an entire population of all possible individual fitness. Hence, search in the fitness landscape is guided by the definition of ecosystems where the species in evolution will interact, in particular by the definition of their own fitness [7]. Therefore the set of points in the landscape is the result of applying an evaluation function to each possible member on the entire population during simulated evolution; as a consequence individuals are rewarded according to their own performance when resolving a specific task.

The fitness of an individual together with the operator's reproduction, crossover and mutation spawn a new generation which corresponds to a specific point in the landscape. Whether this point is closer to the optimal fitness depends on the degree of fitness of the individual. The more fitted is the individual the closer the search is to find an optimal solution. In order to acknowledge the fitness of a population, each individual has to be evaluated during its own lifespan while developing the abilities to solve a particular task. Next, a specialized representation of the robot controller, to be tested, is coded into chromosomes where each locus (or position) takes a finite possible value (allele). This representation is the genotype from where the phenotype will be derived. Often the genotype directly codes into the phenotype; however sometimes an elaborated translation is needed. Initially, a population of random controllers is spawned, and then three GA operators are applied.

## 3 The Evorobot Simulator

EvoRobot* is a software simulator [8] developed by Stefano Nolfi and Onofrio Gigliotta at the Laboratory of Artificial Life and Robotics (ISTC-CNR) in Rome Italy. This robot simulator is a powerful tool for experimenting with the e-puck and the Khepera robots. Additionally, it allows the simulation of individual and collective tasks. This software employs neuronal networks for robot control and genetic algorithms for the optimization of proposed behavior. In order to run the experiments is necessary to configure parameters related to the definition of the simulated world where the robot is to be set. Some of these parameters define: the number, size and position of surrounding walls; number, position and size of food-zones; position and number of lights; and also the number and position of related landmarks. On the other hand, the characteristics of the simulated robot are defined with parameters for various sensors, topology of the neuronal controller, lifetime of the robot, the fitness function and also parameters for the genetic algorithm. Once these parameters are set evorobot tests several robo-controllers using evolution. The weights of the neuronal network on each individual are modified according to an already defined fitness function. Therefore, evolution shapes behavior by analyzing the fitness of both the best individuals and the average of all the individuals on the currently generated population. The final fitness was defined using an incremental approach; thus the simulator had to be recompiled every time the fitness function was modified. Finally, is possible to add new sensors by modifying the open source code of evorobot.

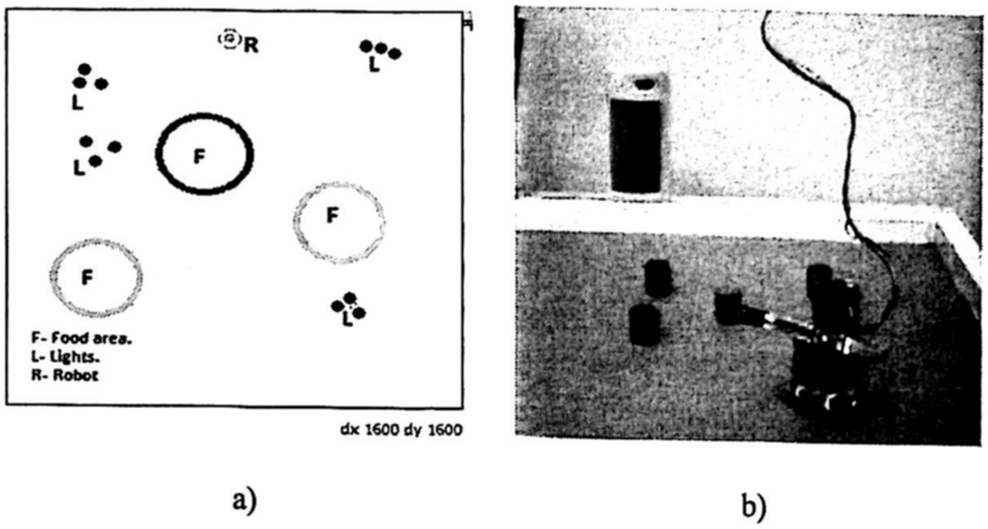

*Fig. 1. a) The setup of the simulated e-puck robot: 3 healthy food zones (F circles), poisonous food as 4 sets of lights (L circles) and 4 walls (squared environment); b) The Khepera setup scenario with poisonous food represented as colored wooden cylinders and a recharging station as healthy food.*

## 4 Robotics Experimental Setup

We have defined a general scenario for our experiments with small variations for testing evolved neuro-controllers in either simulated or real robots. In both cases the testing arena is defined as a squared area. As for the environmental setup in the simulator, available in the world area there are four sets of three lights (poisonous food) and three zones of healthy food (see figure 1-a). On the other hand, in the real world scenario healthy food is recognized using colored tags adhered onto different objects. Poisonous food is implemented as paper colored wooden-cylinders and red leds (light emisor diodes) attached on top of non-colored cylinders; all them scattered around the arena (see figure 1-b). Thus, we use two different kinds of unhealthy food to test the Khepera under different color variations. Next, fitness was implemented rewarding those individuals that avoid cylinders and lights approaching nearby food zones.

$$f = \sum_{i=0}^{1000}(\text{abs}(\text{m}_\text{l} + \text{m}_\text{r})(1 - \sqrt{m_l + m_r})(1 - maxir_i)) - 0.5 \cdot maxls_i + 0.25 \cdot food\_zone\_on_i \qquad (1)$$

where for iteration *i*: $m_l$ and $m_r$ respectively are left and right motor speeds, $maxir_i$ is the highest infrared value. The highest light sensor value is $maxls_i$ and *food_zone_on* takes a binary value depending on the detection of available food.

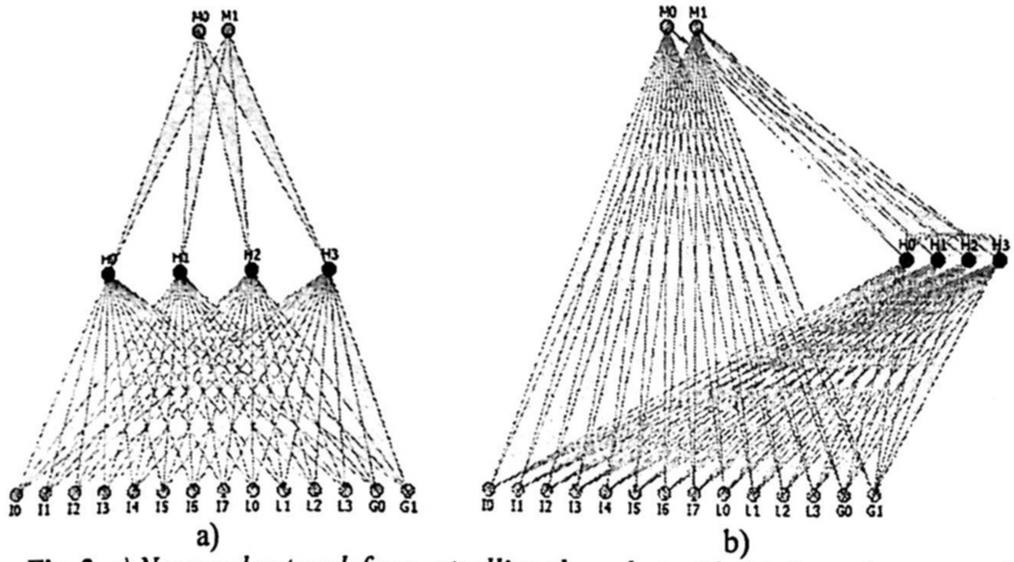
M0 M1
H0 H1 H2 H3
I0 I1 I2 I3 I4 I5 I6 I7 L0 L1 L2 L3 G0 G1
a)
M0 M1
H0 H1 H2 H3
I0 I1 I2 I3 I4 I5 I6 I7 L0 L1 L2 L3 G0 G1
b)

***Fig. 2.*** *a) Neuronal network for controlling the robot with 8 infrared sensors, 4 light sensors and 2 ground sensors; one hidden layer with 4 neurons and 2 neurons in the output layer (corresponding to the motors of the robot);*
*b) Neuronal network with 8 infrared sensors, 4 light sensors, 2 ground sensors; a hidden layer with 4 neurons; 2 neurons in output layer. Also we added recurrent connections in the middle layer and direct connections between the input and output layers.*

The configuration of the neuronal network is as follows. In total the input layer has twelve neurons that correspond to each of the employed sensors (eight infrared sensors, two ground sensors for finding different available food and four light sensors). The hidden layer has four neurons while the output layer has two neurons that represent the robot motors (see figure 2-a). In addition to the above described controller, we evolved a different neuronal network with the same number of neurons for all the layers, but in this case using recurrent connection at the hidden layer and connections between the input and output layers (see figure 2-b). In order to test the non-recurrent neuro-controller on the real robot platform, we export the evolved neural network with its weights into a text format. Thus, in Matlab we re-implemented the neural network to control and process in the Khepera the available infrared, led, and visual information. Next, a standard RS232 interface was used to connect the robot to the computer host. The foraging task for the Khepera robot was set in an arena for running the experiments in a similar fashion to that of Montes-González, et al. [10].

Next, poisonous 'food' was set using wooden cylinders, which were set around the arena. The infrared inputs of the neural networks were taken directly from the Khepera robot. Additionally, red lights simulated poisonous food and the infrared sensors were used for detecting this kind of food. Though, different nodes at the input layer of the neural network were used. Therefore, the infrared sensors, in the ring around the robot body, were used to avoid both kinds of poisonous food. Food areas were simulated as three recharging stations using green or blue colored paper glued in front of external computer speakers. The different inputs of the ground sensors of the neural network were used to determine that a specific colored recharging battery is either located in the image taken from the Khepera robot. Therefore, cylinders of food were treated as obstacles sensed by the infrared sensors, lights as poisonous food that had to be avoided, and healthy food were the recharging stations detected using color segmentation.

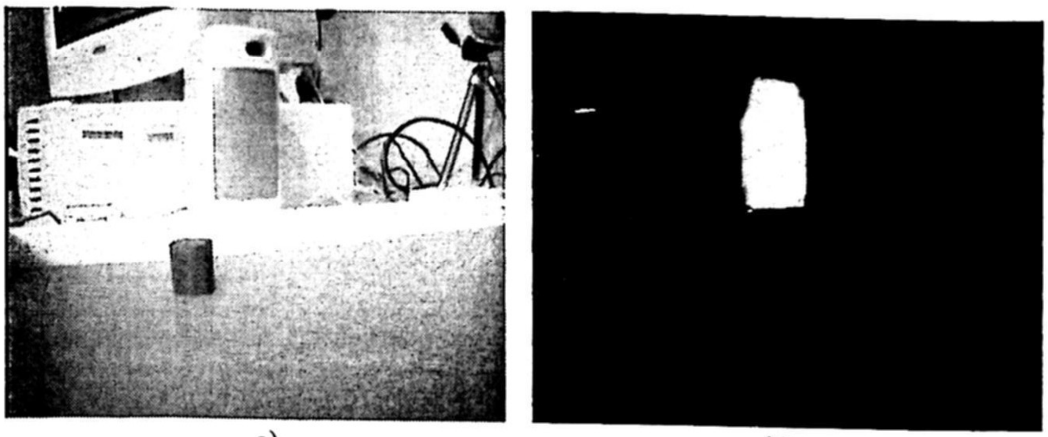

*Fig. 3. a) A real image acquired from the Khepera robot camera. Healthy food isidentified by the green tag glued onto the speaker; b) Object recognition using color segmentation is shown in green.*

The use of visual information from the camera is extremely rich, and needs to be extracted from images taken on-the-fly. A major problem in dealing with visual information is related to illumination changing conditions. A simple way to achieve chromaticity invariance [11], with respect to illumination changes, relies on the conversion of the RGB space into a normalized color space (rgb).

$$r = R/(R+G+B)\ ,\ g = G/(R+G+B)\ , b = B/(R+G+B) \qquad (2)$$

As suggested in [12], this approach can be interpreted as obtaining a simple form of color constancy under various illuminating conditions while preserving directional information of the color vector. The colored object of interest is characterized by its chromaticity, in the normalized color space, using a statistical color distribution. We assume that previously similar objects have already been learned, and that the distributions are Gaussians. Then, we use a statistical band-pass filter for the colored object. Figure 3 shows color segmentation for nutritious food.

## 5 Results

The evolution of two kinds of neuro-controllers was carried out. Evolution was stopped after 30 generations (figure 4). The evaluation of evolved individuals in each generation was carried out after five trials of the best individuals and five trials of random individuals. As a consequence various different aspects were evaluated for the experiments: type of movement, avoiding collisions, finding food zones and the avoidance of lights (see tables 1 and 2).

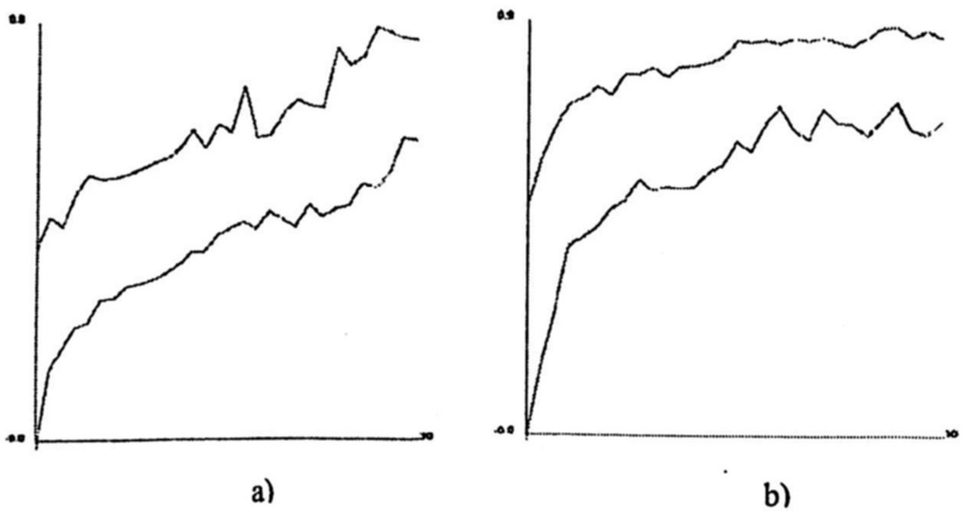

*Fig. 4. a) Evolution after 30 generations of a non-recurrent neuro-controller. The upper curve shows the best individual in each generation and the lower curve shows the average individual fitness; b) The fitness for the evolution of a retro-connected neural network after 30 generations. The upper curve shows the best individual and the lower curve shows average individuals in the population.*

Summarizing, we observe that after the evolution has been stopped a neuro-controller without recurrent connections is able to complete its main goal slower than a neuro-controller with recurrent connections. The evolution of non-recurrent neuro-controllers presents variations in the fitness affecting detection and hence avoidance of lights. On the other hand, recurrent neuro-controllers evolve at a regular pace during initial generations presenting a non-decreasing fitness functions (see figure 4). In general, the more stable reactive non-recurrent controller takes more generations to find a better solution. On the contrary, the recurrent controller finds a solution after few generations and is less reactive to locate objects [9].

**Table 1.** The behavior of the no-recurrent neuro-controller is decomposed into elementary actions. After generation $10^{th}$ the overall behavior of the robot is irregular when avoiding obstacles; but is able to finding food areas, avoiding lights, and moving forward.

| Generation | Movement | Avoiding collisions | Finding food areas | Avoiding lights | Generation | Movement | Avoiding collisions | Finding food areas | Avoiding lights |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Backward | No | Yes | No | 15 | Forward | Yes | Yes | Yes |
| 1 | Circular | No | Yes | No | 16 | Forward | Yes | Yes | Yes |
| 2 | Elliptical | No | Yes | No | 17 | Forward | Yes | Yes | Yes |
| 3 | Elliptical | Sometimes | Yes | No | 18 | Forward | No | Yes | Yes |
| 4 | Circular | Yes | Yes | No | 19 | Forward | Yes | Yes | Yes |
| 5 | Spiral | Yes | Yes | No | 20 | Forward | Yes | Yes | Yes |
| 6 | Spiral | Yes | Yes | No | 21 | Forward | No | Yes | Yes |
| 7 | Forward | Yes | Yes | Sometimes | 22 | Forward | Yes | Yes | Yes |
| 8 | Forward | Yes | Yes | Sometimes | 23 | Forward | Yes | Yes | Yes |
| 9 | Forward | No | Yes | Sometimes | 24 | Forward | Yes | Yes | Yes |
| 10 | Forward | Yes | Yes | Yes | 25 | Forward | Yes | Yes | Yes |
| 11 | Forward | Yes | Yes | Yes | 26 | Forward | No | Yes | Yes |
| 12 | Forward | Yes | Yes | Yes | 27 | Forward | Yes | Yes | Yes |
| 13 | Forward | Yes | Yes | Sometimes | 28 | Forward | Yes | Yes | Yes |
| 14 | Circular | Yes | Yes | Yes | 29 | Forward | Yes | Yes | Yes |

***Table 2.*** *The table summarizes the elementary actions on robot behavior during the evolution of a recurrent neuro-controller. After third generation, individuals complete the foraging task: avoiding obstacles and lights, finding food areas, and moving forward.*

| Generation | Movement | Avoiding collisions | Finding food areas | Avoiding lights | Generation | Movement | Avoiding collisions | Finding food areas | Avoiding lights |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Backward | No | Yes | No | 15 | Forward | Yes | Yes | Yes |
| 1 | Circular | Yes | Yes | No | 16 | Forward | Yes | Yes | Yes |
| 2 | Spiral | Yes | Yes | No | 17 | Forward | Yes | Yes | Yes |
| 3 | Forward | Yes | Yes | Yes | 18 | Forward | Yes | Yes | Yes |
| 4 | Forward | Yes | Yes | Yes | 19 | Forward | Yes | Yes | Yes |
| 5 | Forward | Yes | Yes | Yes | 20 | Forward | Yes | Yes | Yes |
| 6 | Forward | Yes | Yes | Yes | 21 | Forward | Yes | Yes | Yes |
| 7 | Forward | Yes | Yes | Yes | 22 | Forward | Yes | Yes | Yes |
| 8 | Forward | Yes | Yes | Yes | 23 | Forward | Yes | Yes | Yes |
| 9 | Forward | Yes | Yes | Yes | 24 | Forward | Yes | Yes | Yes |
| 10 | Forward | Yes | Yes | Yes | 25 | Forward | Yes | Yes | Yes |
| 11 | Forward | Yes | Yes | Yes | 26 | Forward | Yes | Yes | Yes |
| 12 | Forward | Yes | Yes | Yes | 27 | Forward | Yes | Yes | Yes |
| 13 | Circular | Yes | Yes | Yes | 28 | Forward | Yes | Yes | Yes |
| 14 | Forward | Yes | Yes | Yes | 29 | Forward | Yes | Yes | Yes |

## 6 Discussion

In this paper we have shown a foraging robot task with non-homogenous food optimized by the use of genetic algorithms. The same evolved neural network was used to control both a simulated e-puck and a real Khepera robot. At the two different robot scenarios robots complete the task of approaching healthy food and avoiding poisonous food. In future experiments we intend to allow a predator co-exist with the foraging robot. Therefore, we expect to combine the use of the real e-puck and Khepera robots. Finally, we plan to further extend the "copy-lefted" evorobot simulator.

## References

1. Goldberg, D. E.: Genetic Algorithms, in Search Optimization & Machine Learning. Addison Wesley Longman Inc. (2001)
2. Doncieux, S., Mouret, J.: Behavioral diversity measures for Evolutionary Robotics. In: IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)
3. Mouret, J., Doncieux, S.: Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation (Trondheim, Norway, May 18 - 21, 2009, pp. 1161--1168. IEEE Press, Piscataway, NJ (2009)
4. Deb, K.: Multiobjectives optimization using evolutionary algorithms. Wiley (2001)
5. Nolfi, S., Floreano, D.: Evolutionary Robotics. The MIT Press (2000)
6. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
7. Santos, J., Duro, R.: Artificial Evolution and Autonomous Robotics (in Spanish). Ra-Ma Editorial (2005)
8. Nolfi, S, Gigliotta, O.: Evorobot* User's Manual. ISTC-CNR, Rome, Italy (2002)
9. Nolfi, S., Marocco, D.: Emergence of Communication in Embodied Agents Evolved for the Ability to Solve a Collective Navigation Problem. Connection Science, vol. 19, Issue 1, pp. 53--74 (2007)
10. Montes-González, F. M., Marín-Hernández, A.: Central Action Selection using Sensor Fusion. In: Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04), pp. 289--296. IEEE Press (2004)
11. Sharma, G., Trusell, T.J.: Digital color imaging. In: IEEE Trans. on Image Processing, vol. 6, pp. 901--932. IEEE Press (1997)
12. Swain, M. J., Ballard D.H.: Color Indexing. International Journal of Computer Vision, vol. 7, pp. 11--32. (1991)