Evolving Robot Behavior for Centralized Action Selection

Fernando M. Montes-González[1], and José Santos Reyes[2]

[1]Facultad de Física e Inteligencia Artificial, Universidad Veracruzana
Sebastián Camacho No. 5, Centro, Xalapa, Ver., México
fmontes@uv.mx
[2]Departamento de Computación, Facultad de Informática,
Universidade da Coruña, Campus de Elviña, 15071 A Coruña
santos@udc.es

Abstract. An effective central model of action selection for solving a foraging task has been presented in past papers. The main task is decomposed in behavioral modules that use multiple sensor information fused together in the form of a unified world perception. In turn, the urgency to-be-executed of each behavior is calculated from its internal status and the unified perception of the world. In this model, the behavior with the highest salience is allowed to be expressed through motor commands. However, for selection to occur we assume that behavioral modules have already been learnt; as a consequence it is necessary to have designed these modules at an earlier stage. In this paper, we propose the use of genetic algorithms to nearly optimize behaviors related to travel the arena where the robot is to be set. Furthermore, we propose that by sharing the same topology as the evolved behaviors, backpropagation can be used to train the locating cylinders behavior.

1 Introduction

The Action Selection problem is a recurrent topic in robotics. This problem finds its roots in ethology where it is termed the "behavior switching" or the "decision making" problem. A developing interest in action selection has grown in researchers looking for modeling animal behavior in robots. Therefore, either a set of tasks or a single task has been chosen for building and programming these animal-robots (*animats*). Mostly, these tasks are concerned with the foraging behavior and social behaviors such as flocking and prey-catching among others [1]. Once the main task is set for the robot to be solved, a mechanism for selection is needed. Thus, a variety of action selection mechanisms can be used, these models range from arrow-box diagrams, and Gedanke experiments [2], to complex equations for replicating animal behavior.

Whatever method is chosen, the main task has to be decomposed in algorithmic versions of behaviors that fused together solve the desired task. Next, the action selection mechanism, and the algorithmic behaviors have to be implemented using any available robotic platform. Thus, the design of the robot task using the *animat* approach meets specific needs that the roboticist has to fulfill if the task is to be re-

Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 213-222

solved. Frequently, roboticists focus more on designing either the action selection mechanism or the behaviors that compose the main task. In this paper, we are trying to focus on both the design of the action selection mechanism and the selected behaviors. Thus, we intend to use an evolutionary approach to fine tuning some of the behaviors that the action selection mechanism has to switch. Besides, we also plan to use backpropagation to train another behavior. Finally, a model of Central Action Selection with Sensor Fusion (CASSF) is used to switch between these behaviors.

The development of the experiments below described requires some necessary background on Genetic Algorithms, which is explained in section 2. Later on, in section 3 we explain how we design the behaviors: *cylinder-seek*, *cylinder-pickup*, *wall-seek*, *wall-follow* and *wall-deposit*. Then, these behaviors will be used in conjunction to solve the foraging task set for the Khepera robot. The selection of the behaviors is made using the CASSF model, which is explained in Section 4. Next, section 5 presents the results of integrating the evolved behaviors, and the neural-net trained behavior with some algorithmic behaviors. Finally, in section 6 we provide a general discussion highlighting the importance of these experiments.

2 Evolving Robot Behavior

Several evolutionary methods have been considered by the robotics community for producing either the controllers or the behaviors required for robots to perform their assigned tasks, and survive in their environments. All of them may be taken as variations of a general process whereby each generation of individuals from a population is evaluated, and these individuals procreate according to their fitness. Later on, the population undergoes mutation processes and a new population is somehow selected from the old one to continue with the process. These methods include Genetic Algorithms (GAs), Evolutionary Strategies, Genetic and Evolutionary Programming and Co-Evolution, although in this work we have used GAs [3].

The use of evolutionary techniques for the development of robot control systems commonly relies on the use of neural networks [4]. Therefore, a population of robot controllers is encoded into genotypic representations; then the selection, crossover, and mutation operators are applied to the population in order to produce a new offspring. The decoding of every genotype into individual phenotypes (neural controllers) allows the evaluation of the new offspring by having each individual to live for a limited period of time. This process is repeated until a satisfactory evaluation level is obtained (Figure 1).

In order to find a solution within a search space the genetic operators are applied to move across a convoluted landscape (Figure 2). This landscape is the result of measuring the fitness of all the individuals of the population. In the evolutionary robotics methodology measuring the fitness corresponds to how well the robot performs when evolving a particular behavioral module. Next, the robot is allowed to operate in the environment for an adequate number of steps or lifetime. The fitness calculation implies robot time-consumption; therefore, in the majority of the cases the use of a robot simulator is preferred. In order to minimize the "reality gap" between the simulator and the actuators, the simulator has to introduce different noise-levels in the sensors

and the actuators. As a result an optimal final transference of the controllers from the simulator to the real robots can be obtained [4,5].

Usually the specialized representation of the robot controller to-be-tested is coded into a chromosome, where each locus (or position) takes a finite possible value (allele). This representation corresponds to the genotype from where the phenotype is derived. Often the genotype directly codes into the phenotype; however, sometimes an elaborated translation is needed. Initially, a population of random controllers is spawned, their fitness evaluated, and then the GAs' operators are applied. The *selection* operator chooses to breed, based on the individual fitness, the best individuals in the next offspring by means of operators such as the crossover and the mutation. The selection is made in a probabilistic way; thus, the use of the weighted *roulette-wheel* is a common method of selection. In this method, each slot of the roulette corresponds to an individual in the population, and the size of this slot is proportional to its fitness.

Another way of selecting individuals is the *tournament selection* where new individuals are generated from the result of local competitions. In the competitions the fittest individuals who beat the less fitted; the former are inserted back in the new progeny whether using or not the crossover operator. The preservation of the best individuals, from a previous generation, into the next offspring is called *elitism*. Selection in this way is used to guarantee that the best solution found so far is not lost. The *crossover* is a probabilistic operator, which takes two individual encodings and swaps their contents around a random point along the chromosome (one point crossover). *Mutation* occurs with a probabilistic flip of the bits in the chromosome encoding of the individuals of the new offspring (in general, with a random change of the alleles of the genes). Often, mutating the individual contents of a new offspring causes inferior individuals; though, better individuals will occasionally emerge.

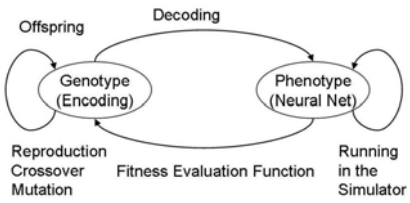

Fig. 1. The new offspring is generated from the genotype of previous robot controllers (adapted from [6])

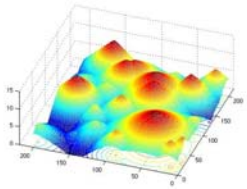

Fig. 2. A fitness landscape generated using a DF1 test problem generator [7]

3 The Design of the Behaviors

In the last decade the use of a commercial robotic platform and the use of a robot simulator have been a popular choice for researchers trying to model robot behavior. One example is the Khepera robot [8], which has been commonly used in evolutionary robotics. The Khepera is a small robot, which has a diameter-long of about 70 mm; the two DC motors control the displacement of the robot on its wheels. This ro-

bot has been equipped with a ring of eight infrared sensors all around the body of the Khepera. Despite many simulators of the Khepera been freeware over the Internet, the use of a commercial simulator has several advantages over freeware software.

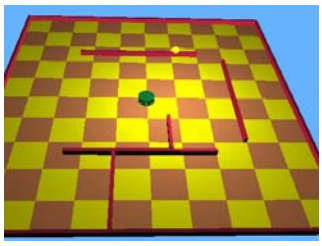

Fig. 3. The *wall-seek* and *wall-follow* behaviors were evolved using the Webots Simulator

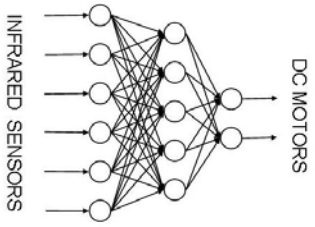

Fig. 4. The behaviors *wall-seek*, *wall-follow* and *cylinder-seek* share the same Neural Network topology

One of the main advantages of using commercial software for robot control and simulation is a complete support of turret attachments. Webots is a 3D robot simulator [9] that widely supports the Khepera robot amongst other platforms. In this work we employed the Webots simulator for developing the follow walls and avoid obstacles behavioral modules (Figure 3). The controller for these behaviors is a fully connected feedforward multilayer-perceptron neural network with no recurrent connections (Figure 4). The topology for the neural network is as follows: 6 neurons in the input layer, 5 neurons in the hidden layer, and 2 in the output layer. The sigmoid transfer function is used at the hidden and the output neurons. The infrared output from the Khepera (0 to 1023) is made binary by using a collision threshold $th_c = 750$. This binary input is fed into the neural network; in turn, the binary output of the neural network is scaled to ± 20 values for the DC motors.

The genetic algorithm nearly optimizes the weights of the neural network, and the use of a direct encoding sets the genotype as a vector **c** of 40 elements. Having a single vector representing every individual of the population, the size of the population is equivalent to all the number of neural networks (to be evaluated) in one generation. Random values are generated for the weights $\mathbf{w}_i$, $-1 < \mathbf{w}_i < 1$, of the $n=100$ neural controllers of the initial population G_0. *Elitism* is used to facilitate the copy of the two best individuals into the next offspring. Four random parents are chosen for $(n/2)-1$ local competitions for the *tournament* selection. The two winners of one local tournament are breed using a random crossover point with a probability of 0.5. On the other hand, the new offspring is affected with a mutation probability of 0.01. Individuals are allowed to run for about 22 seconds in the Webots simulator. In order to let the individuals to start from different locations, and orientations, a supervisor node was set in Webots. Communication of the supervisor with the neural controller, over TCP/IP, facilitated the motion of the Khepera in the fast-speed mode. The world used for running the evolution of both controllers is shown in Figure 3. A world such as this favors avoiding obstacles while traveling close to walls.

For the avoiding obstacles behavior (*wall-seek*), the employed fitness formula (adapted from [10]) for each individual was

$$f_{c1} = \sum^{3000} abs(ls) * (1 - \sqrt{ds}) * (1 - \max\_ir) \quad (1)$$

where *ls* is the linear speed in both wheels, *ds* is the differential speed on both wheels (a measurement of the angular speed), and *max_ir* is the highest infrared sensor value. The use of a formula fitness like this rewards those fastest individuals who travel on a straight line while avoiding obstacles. The plot of the fitness and the average fitness for this behavior, over 11 generations, is shown in Figure 5. Next, the fitness formula employed for the behavior *wall-follow* (adapted from [11]) was as follows

$$f_{c2} = f_{c1} * (tgh)^2 \quad (2)$$

In this formula the tendency to remain close to walls (*tgh*), or thigmotaxis, is calculated as the fraction of the total time an individual is close to a wall. Therefore, a fitness formula such as this selects the individuals that travel near the wall with an avoidance behavior. Figure 6 shows the plot of fitness and the average fitness for the *wall-follow* behavior over 18 generations.

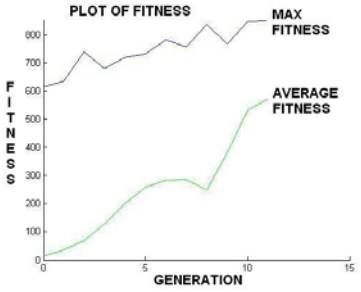

Fig. 5. The evolution of the *wall-seek* behavior is reached after 11 generations

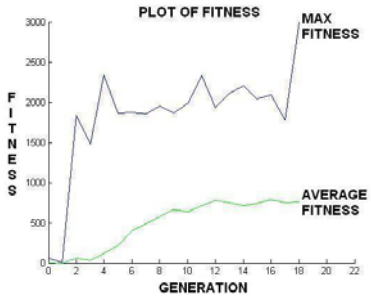

Fig. 6. An appropriate *wall-follow* behavior is obtained after 18 generations

The development of the *cylinder-seek* behavior employed a similar architecture for the neural network than the previous evolved behaviors. However, a set of fifteen patterns was used to train the neural network with backpropagation. These patterns correspond to the main situations the robot finds when the behavior has to be activated. Instead of using a test set of patterns, the trained network was tested on the simulator to assert an adequate generalization. Basically, on this behavior the Khepera, when running around the squared arena, avoids obstacles until is driven close to a cylinder, and then the robot is stopped. Generalization is obtained after 1,184 generations when the total-error drops below 0.02 (Figure 7).

Finally, the *cylinder-pickup* and the *wall-deposit* behaviors were programmed as algorithmic routines with a fixed number of iterations for clearing the space for lowering the arm, opening the claw, and moving upwards the arm. These two actions always have to be carried out in the same sequence, but in a reversed order, to either grip an object or to release the same object. Due to the sequential nature of these behaviors we preferred the use of algorithmic versions of them, rather than trying to

shape those behaviors using supervised learning. Once all the mentioned behaviors were appropriately designed, and evolved, they were implemented on the real Khepera (Figure 8). Due to the simulator properties of introducing noise in the simulated sensors and actuators, and the topology of the neural network acting as a tolerant classifier to noisy inputs; the transference of the evolved behaviors from the simulator to the real robot was made in a straightforward manner. On the other hand, the algorithmic routines for handling the gripper required none modification.

Fig. 7. The Neural Network adequately classifies the 15 training patterns after 1,184 iterations with a total-error below 0.02

Fig. 8. The Khepera robot set in the middle of a squared arena with simulated food (wooden-cylinders)

4 The CASSF Action Selection Mechanism

A centralized model of Action Selection with sensor fusion (CASSF) has already been used for allowing a winning competing behavior to be expressed at the time [1, 12, 13]. A main control loop updates at every step of the simulation sensor readings and motor commands. The different sensors readings from the Khepera, the infrared, the odometry, and the optical barrier in the gripper; all they form the raw sensory information that is to be fed into the model. Next, the raw sensory information takes the form of single perceptual variables that can be used to build a unified perception of the world. Therefore, the use of sensor fusion facilitates the integration of multiple non-homogenous sensors into a single perception of the environment.

The perceptual variables are used to calculate the urgency (salience) of a behavior to be executed. However, not all the variables are equally relevant for a particular behavior. For instance, the searching of a place for releasing a cylinder requires the presence of an object in the gripper. Additionally, behaviors contribute to the calculation of the salience with a busy-status signal indicating a critical stage where interruption should not occur. Therefore, the salience of a behavioral module is calculated by weighting the relevance of the information from the environment (in the form of perceptual variables), and its busy status. In turn, the behavior with the highest salience wins the competition and can be expressed as motor commands that are directly sent to the motor wheels and gripper. Next, we explain how the salience is computed.

The salience is calculated by adding the multiplication of the perceptual variables, by the relevant behavioral weights, to the multiplication of the weighted busy-status. For the foraging behavior, the perceptual variables wall_detector(e_w), gripper_sensor(e_g), cylinder_detector(e_c), and corner_detector(e_r); they form the context vector, which is constructed as follows (**e** =[e_w, e_g, e_c, e_r], e_w, e_g, e_c, $e_r \in \{1,0\}$). Five different behaviors return a current busy-status (c_i) indicating that ongoing activities should not be interrupted. Next, the current busy-status vector is formed as described next, **c** =[c_s, c_p, c_w, c_f, c_d], c_s , c_p , c_w , c_f, $c_d \in \{1,0\}$, for *cylinder-seek*, *cylinder-pickup*, *wall-seek*, *wall-follow*, and *wall-deposit* respectively. The salience (**s**) or urgency is calculated by adding the weighted busy-status (w_b) to the weighted context vector (**e**). Then with $w_b = 0.7$ we have:

$$\mathbf{s}_i = \mathbf{c}_i w_b + \mathbf{e}_i \mathbf{w}_i^e \quad for \tag{3}$$

$$\begin{aligned}
&\text{cylinder} - \text{seek} && \mathbf{w}_s^e = [\ 0.0, && -0.15 && -0.15, && 0.0 \ ], \\
&\text{cylinder} - \text{pickup} && \mathbf{w}_p^e = [\ 0.0, && -0.15, && 0.15, && 0.0 \ ], \\
&\text{wall} - \text{seek} && \mathbf{w}_w^e = [\ -0.15, && 0.15, && 0.0, && 0.0 \ ], \\
&\text{wall} - \text{follow} && \mathbf{w}_f^e = [\ 0.15, && 0.15, && 0.0, && 0.0 \ ], \\
&\text{wall} - \text{deposit} && \mathbf{w}_d^e = [\ 0.15, && 0.15, && 0.0, && 0.15 \ ]
\end{aligned}$$

Behaviors are chosen accordingly to the task that is to be resolved. For instance, the simulated foraging task resembles the food-retrieval of a wall-following rat when afraid to facing a novel environment. The robot basal ganglia proposal of Prescott et al. [14] address more neurophysiological issues related to this work. Whereas for cleaning an arena full of cans, and depositing them close to walls, is solved using a complete evolutionary approach in Nolfi [15]. For our experiments we have decomposed the foraging task in algorithmic behaviors that can be assembled together to show this behavior. Finally, we assume that behaviors were previously learnt, and they can be considered as selected chunks of memory.

The centralized model can be thought as a particular instance of a neural network (Figure 9), and evolutionary learning could be used to set its weights instead of the delta rule. The single-layer feed-forward network (perceptron) has a distribution input layer with four neurons, and the piecewise-linear transfer function at the output layer of five neurons. Using sensor fusion the Khepera raw sensory information is fed, into the neural network, in the form of perceptual variables. Winner-takes-all is implemented at the output of the neural network by letting the highest output to win the competition. In order to contribute on the calculation of the salience, a winning behavior sends a busy signal to the computing output layer of the neural network. A behavior is deselected when its salience is below the strongest salience.

5 Experiments and Results

The foraging task was set in a squared-box for running the experiments. A standard RS232 interface was used to connect the robot to the computer host. The simulated

'food' was made of four wrapped foil-paper wooden-cylinders, which were set in the middle of the arena. Foil paper was used to facilitate the use of the resistivity sensor in the claw of the gripper. The definition of the behaviors, as described earlier, is as follows: *cylinder-seek* is used to travel the arena searching for food while avoiding obstacles, *cylinder-pickup* clears the space for grasping the cylinder, *wall-seek* is employed to locate the nearest wall and avoiding obstacles, *follow-wall* travels next to a proximate wall, and *cylinder-deposit* lowers and opens an occupied gripper.

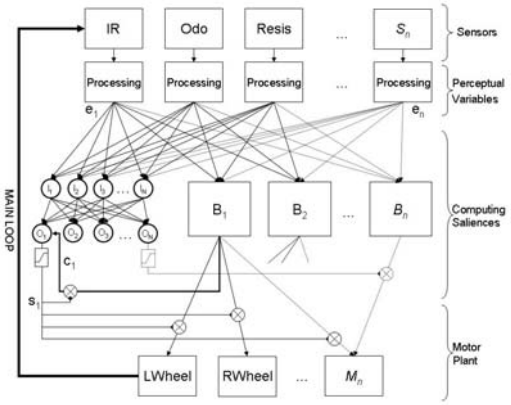

Fig. 9. Perceptual variables (e_i) form the input for the Salience Neural Network. The output selection of the highest salience (s_i) is gated to the motors of the Khepera. Notice the busy-status signal (c_1) from behavior B1 to the output neuron

Commonly, the foraging task is formed by four grasping-depositing bouts of foiled cylinders; thus, one typical grip of a cylinder is shown in Figure 10. Additionally, the next ethogram (Figure 11), and some related statistics (Table 1) resume this behavior. The time resolution of the ethogram is reported in seconds, and seconds and milliseconds for the statistics analysis. Neither of the behaviors was selected before cylinder-seek. However, it took only 0.01 seconds for cylinder-seek to be selected. Next, cylinder-pickup was selected at 3.25 seconds of the total elapsed time (42.64 seconds). Wall-seek was selected at 4.71 seconds, followed by wall-follow at 6.87 seconds, wall-deposit was selected the last after approximately 7 seconds of having initiated the search of a can. The remaining three bouts were repeated in a similar fashion with different search periods. The behavior selected the most was can-seek with 5 times, all the rest with 4 times.

6 Conclusion

In this paper we have shown that non-homogenous behaviors such as wall-seek and wall-follow were the result of using genetic algorithms, and cylinder-seek was obtained using backpropagation. The three behaviors share the same neural network to-

pology, but with different weights, for having the neural network to behave in a different manner. The use of the same circuit with different weights may evidence flexibility in the use of shared circuits for motor control (plasticity). The central model of action selection coupled well with the design of the behaviors, which were developed using Neural Networks, GAs, and backpropagation, together with algorithmic routines designed by hand.

The calculation of the salience in CASSF drives the selection of non-homogenous behaviors for the resolution of a foraging task. Additionally, the use of a busy-status signal accounts for boosting the salience when the behavior has to be maintained, and the disconnection of the same signal favors the interruption of any of the behaviors during the simulation. The occurrence of regular bouts of grasping-depositing cans offers a proof of the correct integration of the non-homogenous behaviors with the model of central selection. Furthermore, due to its detached-from-the-behaviors design, we consider CASSF a robust model of action selection. Besides, adaptability and extensibility is offered, at relatively low cost, by allowing independent modification of both the selection module and the behaviors.

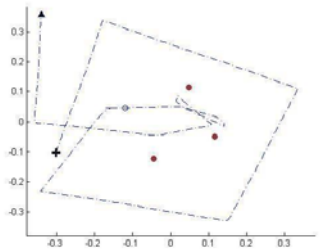

Fig. 10. A regular food-collection bout, the cross indicates the initiation of the behavior

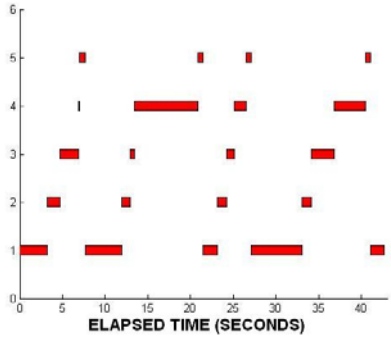

Fig. 11. Ethogram for a typical run of the foraging task

Table 1. Elementary statistics for a representative run of the Khepera with CASSF

Behavioral Elements	Freq	Latency	TotDur	TotDur%	Mean	StdDev	StdErr	MinDur	MaxDur
1. cylinder-seek	5.00	0.01	16.74	39.27	3.35	1.86	0.83	1.54	5.99
2. cylinder-pickup	4.00	3.25	4.59	10.75	1.15	0.20	0.10	1.04	1.45
3. wall-seek	4.00	4.71	6.24	14.63	1.56	1.03	0.52	0.50	2.68
4. wall-follow	4.00	6.87	12.58	29.50	3.14	3.17	1.59	0.14	7.37
5. wall-deposit	4.00	7.01	2.48	5.82	0.62	0.01	0.01	0.61	0.64
6. none	1.00	0.00	0.01	0.02	0.01	0.00	0.00	0.01	0.01
Total	22.00	0.00	42.64	100.00	1.94	1.91	0.41	0.01	7.37

Once the behaviors are evolved or learnt, the next step is the evolution of the matrix $\mathbf{w}^c$. Moreover, the co-evolution of this matrix, which defines the global control

architecture, with the evolved behaviors can simultaneously be made. Finally, as a further step we aim to automatically obtain, by means of computer evolution, the most adequate definitions of the perceptual variables with the later objective of having a reduction in the number of decisions made by the human designer.

Acknowledgments

This work has been sponsored by CONACyT-MEXICO grant #SEP-2004-C01-45726.

References

1. Montes Gonzalez, F. and D. Flandes Eusebio. The Development of a Basic Follow-Behavior within a Distributed Framework. in the 1st IEEE Latin American Robotics Symposium (LARS 2004). 2004. México, D.F., México.
2. Einstein, A. and L. Infeld, The Evolution of Physics. 1966: New York: Simon and Schuster.
3. Holland, J., Adaptation in Natural and Artificial Systems. 1975: University of Michigan Press, Ann Arbor.
4. Nolfi, S. and D. Floreano, Evolutionary Robotics. 2000: The MIT Press.
5. Santos, J. and R. Duro, Artificial Evolution and Autonomous Robotics (in Spanish). 2005: Ra-Ma Editorial.
6. Balakrishnan, K. and V. Honavar. Properties of Genetic Representation of Neural Architectures. in the World Congress on Neural Networks (WCNN '95). 1995.
7. Morrison, R.W. and K.A. De Jong. A Test Problem Generator for Non-Stationary Enviroments. in the Congress on Evolutionary Computation (CEC-99). 1999.
8. Mondana, F., E. Franzi, and I. P. Mobile robot miniaturisation: A tool for investigating in control algorithms. in Proceedings of the 3rd International Symposium on Experimental Robotics. 1993. Kyoto Japan: Springer Verlag.
9. Webots, *http://www.cyberbotics.com*. 2005, Commercial Mobile Robot Simulation Software.
10. Floreano, D. and F. Mondana. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. in From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior. 1994: MIT Press-Bradford Books, Cambridge MA.
11. Bajaj, D. and M.H. Ang Jr. An Incremental Approach in Evolving Robot Behavior. in The Sixth International Conference on Control, Automation, Robotics and Vision (ICARCV'2000). 2000. Singapore.
12. Montes Gonzalez, F. and A. Marin Hernandez. Central Action Selection using Sensor Fusion. in the 5th Mexican International Conference on Computer Science (ENC'04). 2004. Colima, México: IEEE Computer Society.
13. Montes Gonzalez, F. and A. Marin Hernandez. The Use of Frontal and Peripheral Perception in a Prey-Catching System. in the 4th International Symposium on Robotics and Automation (ISRA 2004). 2004. Querétaro, México.
14. Prescott, T.J., F.M. Montes Gonzalez, K. Gurney et al., A robot model of the basla ganglia: behavior and intrinsic processing. Neural Networks (in press).
15. Nolfi, S., Evolving non-trivial behaviors on real robots: A garbage collection robot, in Robotics and Automation System. 1997. p. 187-198.