Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Predation: an Approach to Improving the Evolution of Real Robots with a Distributed Evolutionary Controller

E. D. V. Simões* & D. A. C. Barone**
Laboratório de Robótica Inteligente – Instituto de Informática
Federal University of Rio Grande do Sul
Porto Alegre, 91509-900, Brazil
***simoes@ inf.ufrgs.br, **barone@inf.ufrgs.br**

ABSTRACT

This article describes the implementation of a strategy that selects, destroys, and replaces some individuals of a population of six real autonomous mobile robots. This strategy was called *Predation*. We introduce Predation as a methodology for improving the performance of an embedded evolutionary system developed for the automatic design of robotic controllers. The paper describes how the evolutionary system controls such a small robot population in real time and the effects of predation in avoiding local optimum. It is able to achieve obstacle avoidance behaviour with the robot population evolving while deployed in the field, instead of just using the evolving group to develop an optimum controller for a single robot.

1. INTRODUCTION

In the scope of this paper, the term *predation* represents a technique applied to improving the performance of evolutionary algorithms by destroying some individuals, which are replaced by random configurations to bring more diversity to the population. This technique has been used before in simulation with promising results [1] [2], but in a different context where predator and prey co-evolve to avoid or follow each other. This paper attempts to apply predation for the first time to a population of six real autonomous mobile robots in a different way: in analogy to nature, the robot population can suffer regular attacks of a "predator" that selects the worst ("weakest") robot in the specified generation and destroys ("kills") it, opening space in the population for the migration of new individuals, hence bringing more genetic diversity to the group. This work achieves this by selecting and substituting, after a specific number of generations, the robot with the lowest fitness by a robot with a random configuration (random chromosome).

The predation technique may vary according to different approaches to select, destroy, and replace the individuals of a population [3-4]. For example, only one individual may be selected to be destroyed every time the *virtual predator* attacks, or the attack may destroy a group of individuals. All the individuals of the selected group may be destroyed, or just a smaller random number of them. The destroyed individuals may be replaced by random ones, or by the offspring of the selected breading parents. The frequency of the attacks is another important factor, for enough generations must be left undisturbed to allow the population to recover from the attacks.

2. THE EMBEDDED EVOLUTIONARY SYSTEM

This paper reports the development of a predation strategy that is able to improve an embedded evolutionary system, which controls a group (population) of six autonomous mobile robots. Instead of applying evolution as a solution finder (the traditional approach), here, the robot control system is able to face an open-ended evolution in a mutable environment, since the robots are constantly being modified by evolution to cope with these variations.

This work is based on an embedded evolutionary system described more precisely in [4]. It is able to achieve obstacle avoidance with a population of six autonomous mobile robots that evolve while deployed in the field, instead of just using the evolving group to develop an optimum controller for a single robot [5]. This evolutionary system innovates for it can produce not only a trained robot but also an open-ended evolution, continuously adapting the robot controllers to cope with a variable environment [6].

Figure 1 shows how the robot control circuit interfaces a sensor module, from which it receives the sensor readings, and commands the motor drive module on how to drive the motors. Evolution works with the configuration of the control circuit together with the noise and uncertainly of the environment to synthesise a solution that is, most of the time, different than the expected one, but functional, nevertheless [7].

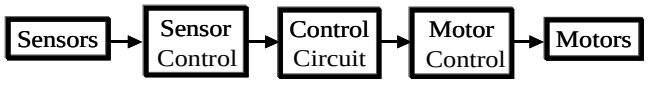

Figure 1 – How the evolvable control circuit fits in the robot architecture.

Differently from other authors that involve simulation in some of the evolutionary phases, the term *Evolutionary System* is applied in this article to describe

an environment where the individuals physically exist and artificially breed and die, to give place to the next generation. For that reason, it is not an evolutionary algorithm, but a real evolutionary system.

Richard Watson *et al*. [8] and the publication of the preliminary results of this work [9] in 1999 provided the first experiments with an Embedded Evolutionary System with real robots. In this work, evolution takes place fully on-board of a population of real robots, working completely independent of external computation or human intervention. Therefore, it is the physical implementation of a genetic system containing the robots as the individuals and a genetic code (bits stored in the RAM memory of the robots) that specifies the configuration of their control device, their speed, and sensor organisation (Morphology). In the scope of this work, the term morphology is defined as the physical, embodied characteristics of the robot, such as its mechanics and sensor organisation. The embedded evolutionary system evolves both the control circuit and the morphology of the robots. Figure 2 shows that the genetic material of the robots can define their control circuit (the configuration of a Neural Network), and the position of the sensors and the precise value of motor velocities.

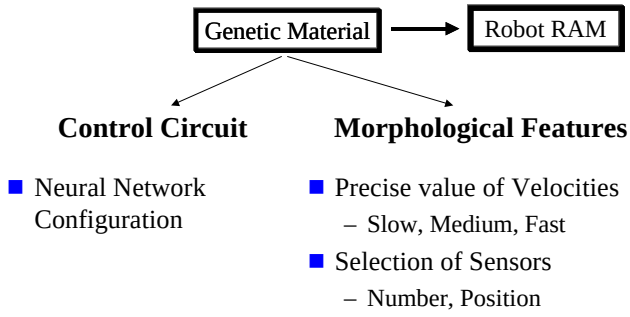

Figure 2 – The genetic material defines the control circuit and the morphological features of the robot.

A relatively simple task that does not involve explicit robot interaction was chosen: collision-free navigation [10]. Therefore, the robots are encouraged to explore the environment while avoiding collisions into the walls, obstacles, or other robots [11]. Collision-free navigation may be trivial for simulated robotic domains, but provides a difficult search space for such a small population, mainly when both robot control and morphology are evolved [12].

2.1 THE ROBOTIC POPULATION

The workspace consists of six autonomous mobile robots working in a 2.50m x 2.50m domain, where they navigate. Figure 3 shows the six robots in their working domain. The robot architecture consists of a round 2-wheeled base (20cm of diameter), containing a Motorola 68HC11 - 2MHz, 128Kb of RAM, bumpers with collision sensors, and eight peripheral infra-red sensors. A RAM neural network, consisting of an n-tuple classifier [13] with 28 x 2-input neurons, implements the evolvable robot control due to its good redundancy [14]. The internal architecture is specified by this genetic code at the control circuit level. This control circuit is implemented within the on-board microprocessor, capable to be reconfigured to produce new generations of more adapted robots. The evolutionary system is not based in an external computer, but is distributed among the robots and coexists with their evolvable controller inside the microprocessor. The distributed evolutionary system of each robot communicates to the others at 1.2Kbps through an embedded 418MHz AM radio.

Figure 3 – General view of the six robots and their working domain.

The neuron contents are directly encoded onto a linear bit-string genotype containing 112 bits (28 neurons times 4 bits per neuron). The physical characteristics are encoded in another 26 bits (10 bits define the speed levels slow and fast and 2 bits are used to enable or disable each one of the 8 sensors). These bits are read from the chromosome in the specified order, the first 112 configure the neuronal controller (the evolving controller), the following 10 the motor control module, and the remaining 16 configure the sensor control module (the robot morphology). The robots work in a cyclic procedure, where they have a working phase, where they try to perform the selected task, and a mating phase, where they reproduce [15]. Figure 4 shows the continuous evolutionary process where the robots are constantly adapting to changes at the environment.

According to the selection criteria, the robots that are more adapted to perform the specified task have more chance of generating descendants. Therefore, the most well-adapted robots survive and spread their characteristics. After many generations, the evolutionary system is able to produce a majority of well-adapted

robots, qualified to work in the environment, and a few unfit robots.

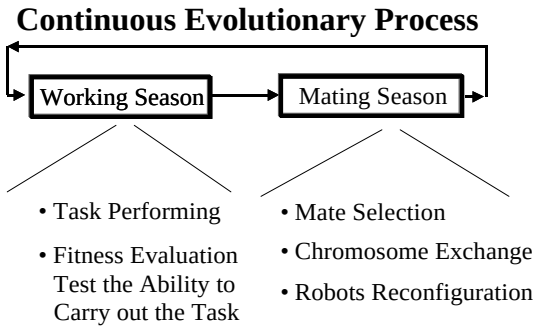

Figure 4 – The continuous evolutionary process.

3. EXPERIMENTS IN SIMULATION

This session presents the experiments that made use of a simulator to produce data much faster than using real robots. A simulator was applicable for it provides an ideal environment, without noise and where the interactions among the robots can be carefully specified. This simulated evolution took much of the complexity of the system away and provided important insights on the specification of the predation strategy to be used with the real robots.

The simulator works by artificially generating all possible sensorial input that a robot can face in its working season and the response of each evolving controller is tested for all these situations and fitness is increased each time the response is correct. Therefore, a perfect condition is created where the best controller always produces the highest fitness score. As all evolving robots are tested for the same conditions, all the noise and uncertainty of the real environment are abstracted away. The simulator permits a clear analysis of the influence of different parameters of the predation strategy. It was built in Borland C++ and run in a Pentium II 200MHz computer.

The experiments in simulation preserved the same small population of six robots. The score of each robot is initialised with 4100 points and the fitness function increases it by one each time the controller output is correct. A total of 256 different situations are evaluated; therefore, the maximum fitness value possible is 4356 points. The experiment tested different mutation rates that are indicated in the charts.

It is important to give enough time to the population so that the attacked robot can recover from the "attack" and its new genetic material can be incorporated in the population. If the attacks occur in less than 5 generations, the attacked robot will not have time to recover and will always be selected as the worst one in the next attack. The necessary time for the attacked robot recover depends on the complexity of the system. In this experiment, the frequency of the attacks is 20 generations. After each working season, the robots are selected to breed according to the selection rule reproduced below:

The robot with the highest fitness survives to the next generation and breeds with all other robots.

This strategy means that the robot with the highest fitness will send its chromosome to all other five robots via the radio. Then each one of the remaining five robots begin a crossover phase, where they combine their own chromosome with the one received from the best robot to produce a resultant offspring. Next, the bits of this resultant chromosome are randomly selected to be mutated (logically inverted) according to the selected mutation rate. Then, the remaining five robots reconfigure themselves with the mutated chromosome and begin the next generation. This tries to make sure that in the next generation the best fitness will be similar to (or higher than) the present one. At least, the surviving best robot has the same chances of repeating its good performance. In the crossover phase, each bit is randomly chosen from the corresponding location in the chromosome of the parents. Then, in the mutation phase, a random number is generated between 0.0 and 100.0 for each bit in the chromosome, and the bit will be flipped each time the generated number is smaller than the mutation rate.

3.1. DISCUSSION OF THE RESULTS

The curves presented in Figure 5 show the effects of the attacks in a robot population that evolved with 0.0% mutation. After the initial improvement resultant form combining the population diversity, the system would stop evolving as shown by the reference curve, where no predation occurred. However, the new genetic material produced by the attacks kept the population evolving up to 4300 points in 300 generations. It can be seen in the figure that the attacks in generations 40, 60, and 120 created a random robot that, combined with the best robot, produced a superior configuration that improved the population performance.

It can be seen in Figure 5 that this strategy improves the system performance if compared to the evolution of a population that did not suffer predation. Figure 6 shows the simulated evolution of four different mutation rates. With the mutation rate of 1%, it reached 90% of the maximum performance in 150 generations. The chart shows that 05% mutation performed worse than 1%, since 05% is not enough to modify at least one of the genes every generation. The mutation rate of 1% has a higher chance of modifying only one or two of the genes, and was better in adjusting a chromosome when most of its genes are already correct.

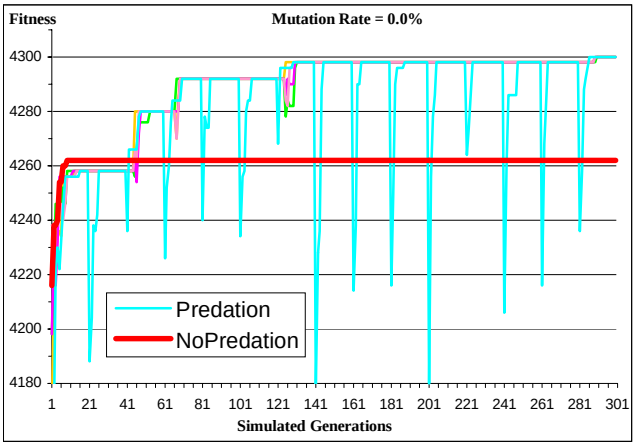

Figure 5 – Simulated evolution of the predation strategy compared to the same experiment without predation.

The predation strategy presented in Figure 6 was able to improve the performance of the evolutionary system in simulation. The whole of predation in this experiment was not to eliminate unfit (weak) robots, since the fitness of the random configurations that substituted the destroyed robot was often worse than the original one. But, even though the resulting random configuration was worse than the destroyed robot, the new genetic material that it contained, combined with the chromosome of the best robot, produced in many occasions a better performance. Therefore, the whole of predation in this experiment was to bring more diversity to such a small population [16].

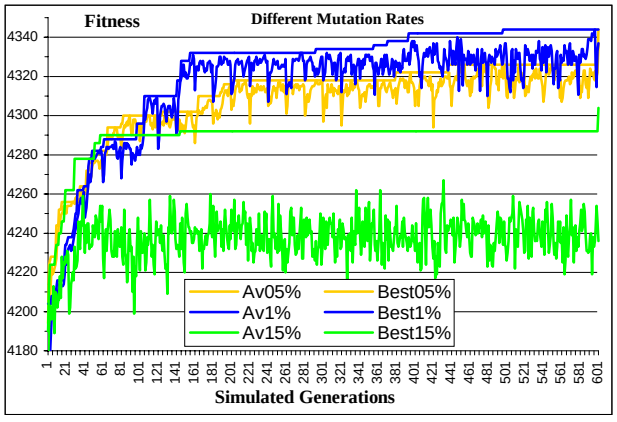

Figure 6 – Simulated evolution of four different mutation rates. **Av** is the average fitness of all the robots and **Best** is the fitness of the best robot.

The instant of the attacks can be identified in the charts by the drop of the fitness of the attacked robot, occurring every 20 generations. This happened because the resulting fitness of the attacked robot, which is reconfigured by a random chromosome, is usually worse than the one that the original robot was producing. Therefore, the average performance of the population usually drops after the attacks. Nevertheless, after a few generations, the new genetic material is filtered by the selection-crossover operators and incorporated in the population, often increasing the average performance.

4. EXPERIMENTS WITH REAL ROBOTS

The aim of this experiment is to test the effects of the predator attack strategy on the real robotic population. This paper shows for the first time the effects of predation in an embedded evolutionary system evolving a neural network controller together with the morphology of the robot. Both the sensor configuration and the speed of the motors are under evolutionary control. This experiment makes use of the most efficient mutation rate and the selection and reproduction strategies that were developed in simulation. It incorporates the developed strategy of predation in an attempt to improve system performance.

In this experiment, the robots were evolved in the same environment presented in Figure 3. The evolutionary system is able to manipulate the bits that control the sensor configuration in the chromosome, plus the bits that control the robot speed levels, and the bits that define the contents of the neurons of the neural network. The number of possible genotypes for the controller is 2^{138}. However, due to the generalization ability of the neural network, many of these genotypes produce the same phenotype. Therefore, the search space is actually smaller. The neural network is connected to eight infra-red sensors. Its architecture has four groups of neurons (discriminators) [13] with seven 2-input neurons, each neuron containing four bits. Therefore, the neural network controller provides four commands to drive the motors: Front Fast (*FF*); Turn Left (*TL*); Turn Right (*TR*); and Front Slow (*FS*).

This experiment uses a very simple fitness function in order to prevent biasing evolution towards a pre-conceived solution. Rule 3 punishes the robots that keep turning for more than 15 seconds, encouraging them to move forward. Rule 4 does not punish the robots that are turning, for they may be attempting to avoid an obstacle when the collision occurred. The selected fitness function for this experiment is:

1- Start with 4096 points;
2- Reward: increase fitness by 10 points every 1 second of movement;
3- Punishment: decrease fitness by 30 points for every time command is not *FF* or *FS* for more than 15 seconds;
4- Punishment: decrease fitness by 10 points for every collision if command = *FF* or *FS*.

The maximum that a robot can score if it does not crash during the 60 seconds of the generation is 4696 points. In this experiment, the robot population suffered

regular attacks (every 10 generations) of a "predator" that selected the robot with the lowest fitness in the specified generation and substituted it by a random one, bringing more genetic diversity to the group. Mutation was set at 1%.

4.1. DISCUSSION OF THE RESULTS

Figure 7 shows the fitness values of the six evolving robots. Since this experiment used a non-biasing fitness function, many different solutions were produced and evaluated by evolution. An interesting point is that maximum fitness was obtained from the first generation. These happened because some robots produced in the first generations developed a form of wall-following behaviour, where they tried to stay close to the walls or obstacles, but keeping a safe distance, so they did not collide with them. This could produce maximum fitness in some generations, but as it can be observed in the chart, one robot could have very distinct performances with the same controller from one generation to the other. They could avoid colliding into the walls, but suffered collision from other robots, so that in some generations their performance dropped substantially. This configuration nevertheless spread quickly through the population and a particular event started to happen when two robots with this configuration started to walk around each other, each considering its fellow robot a wall to be followed. They kept spinning around, crashing upon everything in their way. This was responsible for the drop in performance near generation 13. The population ended up converging to this unstable solution, a local optimum, and could reach a better configuration only because of a predator attack that brought in new genetic material.

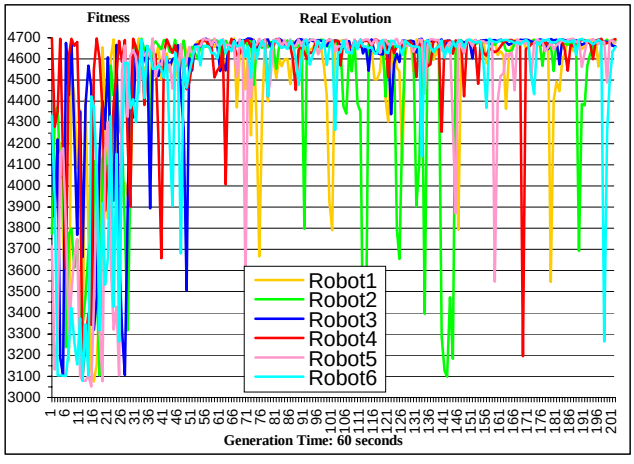

Figure 7 – The population was initialised with a random controller and morphology.

The third attack of the predator, in generation 30, resulted in a robot that could for the first time use one sensor in the front and produced a more steady performance. This can be observed in the chart, where the fitness of each robot started to improve. This happened because, although this configuration did not produce such high scores as the previous wall-following behaviour, its average result was more consistent, and the average fitness of the population increased considerable between generations 30 and 40.

This experiment illustrated the power of the developed predation strategy in providing more diversity when the population was trapped in a local optimum. The new genetic material it supplied in the first thirty generations was essential to allow the population to explore more widely the fitness landscape. The disadvantage of this strategy is that it never allowed the average of the population fitness to reach the maximum score, since a random robot was introduced every 10 generations, causing the performance of the population to drop.

As the number and position of the sensors and the speed levels of the motors are under evolutionary control, not only the control circuit is produced, but also the physical characteristics of the robots can change into different configurations according to the complexity of the environment. In addition, the designer can fix the number of sensors, for example, and let evolution decide where they should be placed. The most successful configurations according to the sensor positions that were observed in the experiments are:

a) Configuration 1 – One sensor in the front;
b) Configuration 2 – Two sensors, one in the front and a lateral one;
c) Configuration 3 – Three sensors, one in the front and one in each side of the robot.

It was observed that, in a simple environment containing few obstacles, all three configurations coexist "peacefully", because it does not present enough selection pressure and the fitness of all three configurations are roughly the same after a short working season (short evaluation time). The longer the lifetime though, the greater the number of opportunities to distinguish between a more efficient configuration and an ordinary one and the robots with more sensors positioned in the right places are more likely to succeed. When more obstacles are added and the environment becomes more complex, the competition is tougher and the configurations with more resources gradually lead the less-adapted ones to "extinction" from the population.

5. CONCLUSION

Predation is a powerful strategy to prevent the population from getting stuck into local optimum, since it introduces new genetic material that may help the

population to crawl down the slope and explore new possibilities in the fitness landscape [18]. Getting stuck in local optimum is an intrinsic problem of most real evolutionary systems, once it is very difficult, and some times impossible, to know from the point of view of the evolving individuals if the population can be improved even more, or if the optimal solution was actually achieved. With the developed predation strategy, the population can count with a steady supply of new genetic material to bring in more diversity, even after it completely converged to a local optimum.

The results obtained with the help of the simulator were essential in providing vital insights on developing new strategies, such as predation, that improved considerably the performance of the system. The simulator also made possible rapid evaluation of different parameters such as different mutation rates, reproduction and selection strategies.

The developed evolutionary system, helped by predation, succeeded in evolving the real robots, initialised with random controllers and morphologies. The experiments demonstrated that this embedded evolutionary system was able to successfully evolve a neural network controller together with the morphology of the robots in real time in the real world. It produced a satisfactory collision-free behaviour in average after 200 generations of 60 seconds.

6. REFERENCES

[1] Nolfi, S. and Floreano, D., "Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution?" *Artificial Life, v. 4, n. 4, pp. 311-335, 1998*.

[2] Chainbi, W., Hanachi, C., and Sibertin-Blanc, C., "The multiagent prey-predator problem: A petri net solution. *In Proceedings of the IMACS-IEEESMC Conference on Computational Engineering in Systems ApplicationLille, France, July 1996, pp. 692-697, 1996.*

[3] Hartono, P.. and Hashimoto, S., "Migrational GA that Preserves Solutions in Non-Static Optimization Problems", *In IEEE International Conference on Systems, Man and Cybernetics, Tucson-AZ, USA, October, 2001, pp. 255-260, ISBN 0-7803-7089-9.*

[4] Simões, E. D. V. and Dimond, K. R., "Embedding a Distributed Evolutionary System into a Population of Autonomous Mobile Robots", *In IEEE International Conference on Systems, Man and Cybernetics, Tucson-AZ, USA, October, 2001, pp. 1069-1074, ISBN 0-7803-7089-9., 2001.*

[5] Floreano, D. and Mondada, F., "Hardware Solutions for Evolutionary Robotics", *In: P.Husbands and J-A.Meyer, editors, Proceedings of the first European Workshop on Evolutionary Robotics. Springer Verlag, Berlin, pp. 137-151,* 1998.

[6] Jakobi, N. and Quinn, M., "Some problems (and a few solutions) for open-ended evolutionary robotics", *In: Proceedings of the First European Workshop on Evolutionary Robotics: EvoRobot'98, P Husbands and J-A Meyer (eds) Springer Verlag, 15p.*, 1998.

[7] Thompson, A. and Layzell, P., "Analysis of unconventional evolved electronics", *Communications of the Association of Computing Machinery, v. 42, n. 4, Yao, X. (Ed.), pp. 71-79,* 1999.

[8] Watson, R. A., Ficici, S. G., and Pollack, J. B., "Embodied evolution: Embodying an evolutionary algorithm in a population of robots", *In Angeline, P., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, Congress on Evolutionary Computation, pp. 335 - 342. IEEE Press,* 1999.

[9] Simões, E. D. V. and Dimond, K. R., "An Evolutionary Controller for Autonomous Multi-Robot Systems", *In IEEE International Conference on Systems, Man and Cybernetics, Tokyo, Japan, October, 1999, v.6, pp. VI596-VI601, 1999, Invited Paper*.

[10] Chavas, J., Corne, C., Horvai, P., Kodjabachian, J., and Meyer, J. A., "Incremental Evolution of Neural Controllers for Robust Obstacle-Avoidance in Khepera", *Proceedings of the First European Workshop on Evolutionary Robotics - EvoRobot'98, Apr. 1998, Paris, France, Husbands, P. and Meyer, J. A. (Eds.), Publisher: Springer Verlag, pp. 227-247, 1998.*

[11] Seth A. K., "Interaction, uncertainty, and the evolution of complexity", *Proceedings of the Fourth European Conference on Artificial Life, Husbands, P. and Harvey, I. (Eds.), MIT Press, pp. 521-530, 1997.*

[12] Pollack, J. B., Lipson, H., Ficici, S. G., Funes, P., Hornby, G. S., and Watson, R. A., "Evolutionary Techniques in Physical Robotics", *Proceedings of the Third International Conference on Evolvable Systems - ICES 2000: From Biology to Hardware, Publisher: Springer , pp. 175-186, 2000.*

[13] Rohwer, R. and Morciniec, M., "A theoretical and experimental account of n-tuple classifier performance", *Neural Computation, v. 8, pp. 629-642, 1996*.

[14] Ludermir, T., Carvalho, A., Brage, A., and Souto, M., "Weightless neural models: a review of current and past works", *In Neural Computing Surveys, v. 2, pp. 41-61, 1999.*

[15] Bäck, T., "On the behavior of evolutionary algorithms in dynamic environments", In D. B. Fogel, H. P. Schwefel, Th. Bäck, and X. Yao, editors, *Proc. Second IEEE World Congress on Computational Intelligence (WCCI'98), Anchorage AK, May 4-9, IEEE Press, v. 1, pp. 446-451, 1998*.

[16] Ficici, S. G. and Pollack, J. B., "Effects of Finite Populations on Evolutionary Stable Strategies", *Proceedings of the 2000 Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, USA, Whitley, L. D., Goldberg, D., Cantu-Paz, E. et. al. (Eds.), Publisher: Morgan-Kaufmann, ISBN: 1-55860-708-0, pp. 927-934, 2000*