

# A-Life Based Optimisation of Non-Stationary Dynamical Systems

M. Annunziato<sup>1</sup>, C. Bruni<sup>2</sup>, M. Lucchetti<sup>2</sup>, S. Pizzuti<sup>1</sup>

<sup>1</sup>ENEA – Energy, New technologies, Environment Agency, ‘Casaccia’ R.C.

Via Anguillarese 301, 00060 Rome Italy

Phone: +39-06-30484411, Fax: +39-06-30484811

email: {mauro.annunziato, stefano.pizzuti}@casaccia.enea.it

<sup>2</sup>University of Rome ‘La Sapienza’

Department of Computer and Systems Science

Via Eudossiana 18, 00184 Rome Italy

Phone: +39-06-44585938, Fax: +39-06-44585367

email: {brunic, lucchetti}@dis.uniroma1.it

## Abstract

In this paper, we develop an intelligent system to approach dynamical optimisation problems emerging in control of complex systems. In particular our proposal is to exploit the adaptivity of an artificial life (alife) environment in order to achieve "not *control rules* but autonomous *structures able to dynamically adapt and to generate optimised-control rules*". The basic features of the proposed approach are: *no intensive modelling* (continuous learning directly from measurements) and capability to follow the *system evolution* (adaptation to environmental changes). The suggested methodology has been tested on the simulator of a classical testbed in dynamical systems experimentations: the Chua's circuit. We supposed not to know the system dynamics and to be able to act only on a subset of control parameters, letting the others vary in time in a random discrete way. We let the optimisation process searching for the new best value of performance, whenever a drop due to changes in fitness landscape occurred. We present the most important results showing the effectiveness of the proposed approach in adapting to environmental non-stationary changes by recovering the optimal value of process performance.

## 1. Introduction

Last decades showed an increasing interest in studying the basic features of complex systems, i.e. all those systems whose non-linear dynamics depend on a great number of strictly related variables, in such a way that the composition of local rules gives rise to a global unpredictable behaviour. This is what happens in every living system, and lots of studies consequently focused on understanding the main characteristics of life.

Researches in this field have culminated with the introduction of a good deal of computer-based simulators of real biological contexts, grouped under the name of *artificial life* [6][7]. An a-life context is an artificial environment where each individual can be regarded as an autonomous agent that has certain computational capabilities and can also interact both with its surroundings and with other similar agents.

Even if there's not a formal theory in a-life research field, it's possible to point out the principal common features. The basic concept in a-life simulations is to program the whole system in order to allow the development of an *emergent behaviour*. Emergence is the concept of some new phenomenon arising in a system that wasn't in the system's specification to start

with [4]. Owing to this, the a-life approach is usually defined with the term *bottom-up*. This goal is achieved by considering the genetics of evolution (the genotype) indissolubly linked with other life aspects, such as interaction, competition, co-operation, etc [9].

At the present, artificial life (or *alife*) is used mainly to study evolution problems, but in our opinion it has the potential to generate information structures that are able to develop an evolving *local intelligence*. With the term *local intelligence* we refer to an intelligence strongly connected to the environmental context (*the problem*) we need to solve[1].

In problems ranging from traffic regulation to energy process control and optimisation, the not-adaptive classical approaches are not effective to solve the problem over time. The not-controlled variables, the process aging, the unforeseeable effects caused by human errors, the evolution of the process, in most of cases require changes in the basic model or in the control objectives, or even in the whole strategy. To reach the goal of evolving structures, a continuous learning of the optimisation system from the environment is necessary but doesn't suffice, and the ability of the system to change its internal structure is

required. We need information structures able to *evolve* in parallel with the process we are modelling[10].

## 2. The evolutionary control approach to dynamic optimisation of complex processes

When studying complex processes, the problems we have to face, are mainly linked to the incomplete information we can get about those systems. Both in control and optimisation, in fact, we need to refer to models representing the observed dynamics. Having no possibility to obtain a suitable mathematical model, we have necessarily to refer to measures-based modelling methodologies, such as expert systems, neural predictors or process on-line simulators.

These methodologies are surely useful for a wide fraction of industrial requirements, but, when applied to dynamical complex processes, they show limitations, summarised in the following items:

- need of strong modelling and/or long training;
- need of heavy knowledge from operators;
- fixed rules for all process life.

The first two problems descend from the above-mentioned modelling. The last one is the most serious problem for traditional methods based on learning: being based on fixed optimisation rules, they don't take care of the evolution of the process during its life.

In this paper we develop a smart adaptive technique, named *evolutionary control*, oriented to optimisation and control of complex processes. Our final goal is to test the approach we are about to describe on a non-stationary dynamical system.

The basic features of the methodology we propose are:

- *no intensive modelling* (progressive training and updating directly from measurements);
- capability to follow the *process evolution*.

The essence could be synthesised by the sentence: "not control rules but autonomous structures able to

*dynamically generate optimised-control rules*". In the construction of the ES, in fact, the knowledge of the operators is verbally transferred to the ES builder. In our proposal, the process knowledge is obtained directly by the system through measurements observation, and it's used to update a dynamic model of the process itself, which we call *performance model*.

The implementation of this idea is described in figure 1. The basic concept consists in the realisation of an artificial environment that *lives* in parallel with the process and that asynchronously communicate with it, in order to dynamically control and optimise it. We suppose to always measure from the process its current regulations and the related value of an observable quantity which we call performance and which represents the objective function of our optimisation. In this way measurements are composed by both process variables and performance. The system continuously gets measurements from the process and provides the process back with the control actions.

The main blocks of the evolutionary control (fig. 1) are the alife environment and the performance model. The first one is an artificial environment composed by individuals able to find the optimal solutions. The second one is a model of the process performance used by the alife environment in order to provide its individuals with a fitness value. These blocks are described in the following paragraphs.

The final control actions are the average between the best solution achieved by the alife environment and the current regulations. This is because we wish smooth transitions among different states.

Each time a new measurement is acquired the performance model is updated with it (*continuous learning*) and a new individual, representing the new experimented/observed process condition, is inserted in the artificial environment.

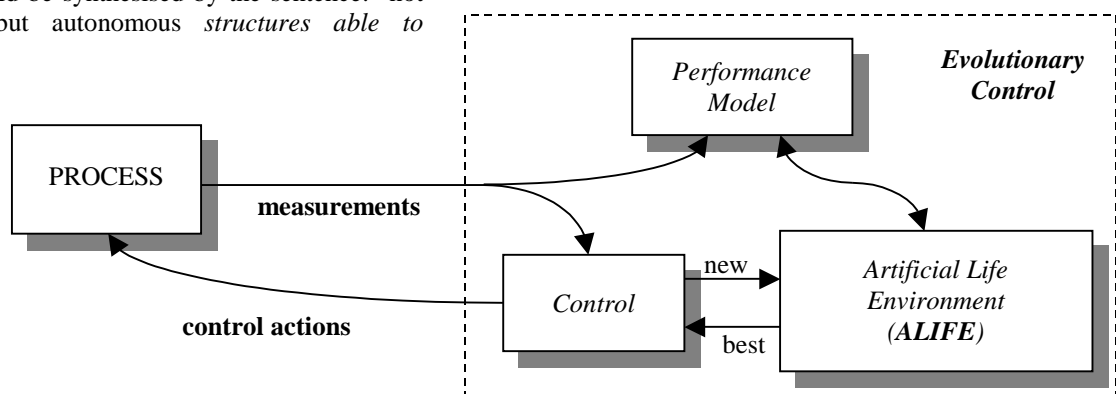


Figure 1 - Scheme of the evolutionary control approach.

Thus the system is continuously updated, it follows the process not-monitored changes and drives the evolution towards better performances. Of course, at the beginning the system is not able to give any suggestion but it only learns from the process measurements. The artificial environment starts being active and giving right suggestions when the performance model is trained.

### 3. The ALIFE environment

The artificial environment implemented derives from the *Artificial Society* approach illustrated in Annunziato et al. [3]. This approach has been tested for the optimisation of a static well known problem, the Travelling Salesman Problem, where it has reached the optimal value for the 30, 50 and 75 towns [3]. The alife context is a two-dimensional lattice (*life space*) representing a flat physical space where the artificial individuals (or *autonomous agents*) can move around. One individual is initially constituted by one single cell, but it can grow along life composing a moving wire (see fig. 2). The length of the wire depends on the individual *energy*. Every iteration (*life cycle*), the individual moves in the life space and, in case of meeting with other individuals, interaction occurs. Each agent has a particular set of rules that determines its interactions with other agents. The three types of possible interactions between agents are combat, mating and disregard. Combat results in the exchange of internal resources between agents (the loser gives a fixed amount of its energy to the winner), and mating results in an offspring whose genome is a combination of those of the parents. Agents can also self-reproduce via haploid mutation. If the two interacting agents have not enough energy to fight or reproduce, they ignore each other and keep on moving.

The agent movement is the result of the composition of a deterministic component and a random component in order to define the curvature of the individual trajectory in the 2D lattice. The dynamics of the individuals are also influenced by the space size (*xsize* and *ysize*) and the *border* type. The border can be open, closed or toroidal. The last has given the best results. In this configuration, when an individual touches a side of the space it enters from the other side.

We suppose to periodically acquire a set of measurements on the real process (*measurement cycle*), to calculate the current value of the actual *performance* and to provide such an information to the control system. The performance is the target of the optimisation and it is derived from measurements.

At every cycle of measurement, a *new* individual is built on the base of the measured values and inserted in the

environment with a starting value of energy (*inlet energy*). A measurement cycle typically corresponds to several life cycles (10-1000). This individual is considered as the ancestor of a new family or *lineage*. All the offspring generated by this individual are marked with the same code of lineage.



Figure 2 - The ALIFE environment

Reproduction is limited by a maximum permitted number of individuals in the space. In order to avoid population saturation this number is chosen quite higher than the natural fluctuations of the population dynamics. Three blocks compose the data structure of each individual: the *genotype*, the *information* and the *status*. The first one includes a collection of behavioural parameters regarding dynamics, reproduction and interaction. The *information* block includes a series of parameters related to the process to control: the regulation and measurement values; both the information and the genotype don't change during the individual life. The *status* parameters include dynamics and structural parameters (position, direction, curvature, wire description), age, energy and performance values. These parameters change during the individual life.

An important feature that makes us think of this strategy as the right one is the finiteness of the individual life. The optimisation, in fact, succeeds in keeping itself updated on the evolution of the process by continuously renewing the population on line. This is allowed by an *ageing* mechanism, according to which each individual dies after a fixed number of life cycles (*average life*).

The performance is updated using an external problem-specific model. This is due to the possible changes in the unknown variables of the process not represented in the genotype. For this reason the performance variable is located in the status block.

We can summarise the main issues of alife approach as follows:

- **biodiversity** – the periodic introduction of new individuals lets the algorithm not to converge towards a local optimum, but several search path

remain active in order to allow a deeper exploration of the regulations space;

- **evolution** – the artificial environment is able to evolve in parallel with the process, continuously renewing all the configurations among which the proposed optimal control set is chosen;
- **adaptivity** – owing to the above-mentioned evolution and biodiversity, the system is able to quickly react whenever an unforeseen change in fitness landscape occurs. This property has been tested in the experimentations we propose.

#### 4. The Performance Model

The performance model represents our knowledge about the process we have to control. The task of the performance model is to provide the newly generated individuals of the alive environment with a performance. The current implementation has three main blocks described in the following figure:

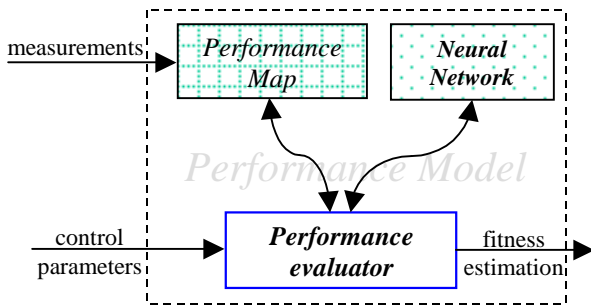


Figure 3 – The Performance Model

Firstly, we have to notice that all the external input parameters we need are process measurements.

The performance evaluator has as input the values of the control parameters and returns an estimate of the corresponding value of the fitness function.

The performance evaluator refers to a twofold structure, composed by an *a-priori model* of the process (we chose a multi-layer neural network) and a table of measures, which we call *performance map*. The control parameters are discretized and the values we obtain are intended to be the performance table indexes.

The performance map accomplishes two specific tasks:

- **consulting**: it stores the process measurements, by means of a discretization of the control parameters, creating in this way an archive of the most recent states of the process; in this sense we can say that it represents a long-term memory of the process history;

- **updating**: it is continuously updated with new measures, keeping information on the plant evolution. The refresh time of the whole map is unavoidably long, because it depends on the exploring strategy of the regulations space; however we have to notice that the updating of the most visited zones occurs by a short refresh time. We can consider this mechanism as a *focus of attention* concept. The best individuals selected by the environment determine the zones of the map where the focus currently is.

We can say that the performance map stands for the *current representation of the system knowledge*.

In evaluating the performance of a given set of control parameters (*regulations*), we have two possible cases, depending on the cell pointed by their discretization: if that cell is already filled with a **measure** then we will consider this value as the estimate of the performance we are looking for; if the input refers to an empty cell then we'll use an *a-priori* model of the process; in this first application, we used a single-layer feed-forward **neural network**.

Clearly, at the beginning the number of empty cells is too high with respect to the filled ones. In this case we should have too often to refer to the *a priori* model, which loses validity in a short time, because of the process evolution. This consideration brought us to introduce an **interpolation** mechanism, according to which in case we point to an empty cell, we have to search for an already measured value in its neighbourhood, within a fixed range. If we find a measure, then we'll interpolate the neural prediction related to the starting empty cell with that value; otherwise we'll consider as performance evaluation the neural prediction.

In figure 4 we show this situation with two control parameters indexing the X and Y axis of the table.

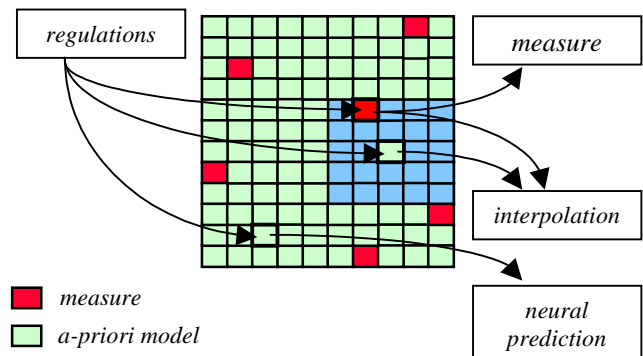


Figure 4 – The Performance Evaluator

In particular we chose a linear interpolation rule,  $K(d) \cdot M_f + (1 - K(d)) \cdot NN$ , where  $M_f$  is the measure found in the neighbourhood of the empty cell,  $NN$  is the

neural estimate of the starting point performance and  $K(d)$  is a weight coefficient, exponentially decreasing with the distance between  $M_f$  and  $NN$  in the parameters space.

## 5. Application to a reference dynamical system: the Chua's circuit

The Chua's circuit [5][8] has been considered for long time as a paradigm in studying the onset of chaotic phenomena. This is mainly due to the possibility to get a wide range of typical chaotic behaviours as well as the simplicity of its design and implementation. The general scheme is given below.

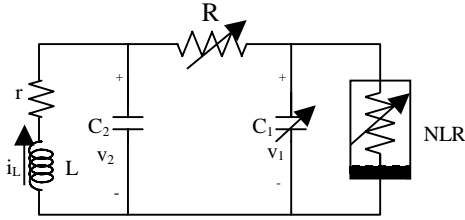


Figure 5 – The Chua's Circuit

The Chua's circuit is an oscillator loaded with a non-linear component (NLR), which we chose with piecewise linear current-voltage characteristic. We simulated this device numerically solving its adimensional state equations:

$$\begin{cases} \dot{x} = \alpha[y - x - f(x)] \\ \dot{y} = x - y + z \\ \dot{z} = -\beta y \end{cases}, \quad (1)$$

where

$$f(x) = bx + \frac{1}{2}(a-b)(|x+1| - |x-1|) + \frac{1}{2}(b-m)(|x+f| - |x-f|),$$

$x \propto v_1$ ,  $y \propto v_2$ ,  $z \propto i_L R$ , and the four control parameters are  $\alpha$ ,  $\beta$ ,  $a$ , and  $b$ . The remaining two parameters ( $m$  and  $f$ ), relate to the intrinsic characteristics of the non-linear components in NLR and so we supposed not to act on their value.

This simulator has been considered as the system to control. Now we just need to define a criterion for the definition of fitness function.

Since we want to carry out a parametric control, the fitness function has not to depend on the initial conditions, but only on the dynamical state of the system, that is on its parameters. The root mean square

(*rms*) of the signal fulfils this requirement. Consequently, we let the system evolve in time just enough to consider the dependence on initial conditions extinguished and then we computed the RMS of the signal. Thus we defined the following normalised fitness function to maximise:

$$\mathfrak{J}(\alpha, \beta, a, b) = 1 - \frac{|\theta - rms(\alpha, \beta, a, b)|}{\theta} \quad (2)$$

where  $\theta$  is a threshold RMS reference value and  $rms(\alpha, \beta, a, b)$  is the RMS of the signal referring to the current values of the parameters  $\alpha, \beta, a$  and  $b$ . We chose the following limitations for the control parameters:

$$\alpha \in [6, 11], \beta \in [12, 18], a \in [-2, -1.05], b \in [-0.95, -0.05]. \quad (3)$$

In those ranges they define a hypercube in the regulations space, such that the system exhibits all its possible behaviours.

## 6. Experimental results

The presented technique is addressed, as we said above, to applications in real complex processes. In real systems it's very hard to build effective models and in general we have no model at all. Our first aim is therefore the control of the circuit, getting rid of the *a-priori* modelling. This consideration brought us to a first class of experiments, planned to demonstrate the effectiveness of the proposed approach aside any kind of modelling. Moreover, in real applications we have to suppose not to be able to act on every parameter that influences the dynamics. Hence we carried out a second class of experiments, where we considered one parameter ( $\alpha$ ) of the process as an unknown disturbance varying in time, giving rise to a dynamic performance function, and we demonstrated that the optimisation system is able to compensate the performance drops due to the non-stationary changes of fitness landscape. In the next subsections we will analyse these two classes of experiments.

### 6.1 A-priori model knowledge vs. on-line model building

As we said above, we demonstrated the control system is able to drive the process to optimal performance aside the *a priori* modelling we consider in the *performance model*. To achieve this consideration, we firstly evaluated the necessity of a high precision in neural predictions. Hence we considered the neural estimate affected by a random increasing noise (10%, 50%,

100% of the nominal best value of the performance). In fig. 6 we compared the resulting best performance trend (dashed line) with the one we have in the case of precise network (continuous line).

The most relevant result is that the optimal performance is reached also in the case of 100% random error on the neural prediction, even if we lose in efficiency.

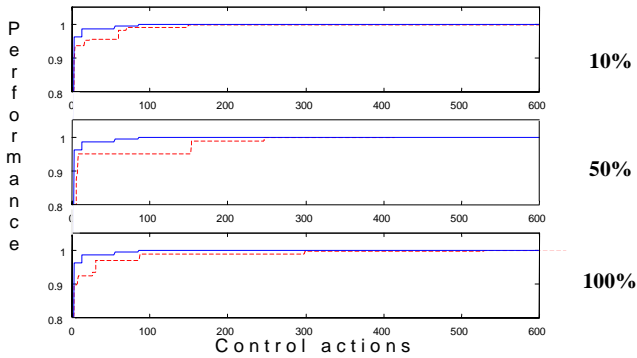


Figure 6 – Noisy neural model

This noisy neural network is still a model of the process, although very rough. In real applications such as control of complex processes, we have often to deal with the complete absence of any kind of model.

Hence we arranged another experiment, where we took the neural network off, substituting it with a random generator. Thus the performance model is but the performance map. The results we found are graphed in fig. 7.

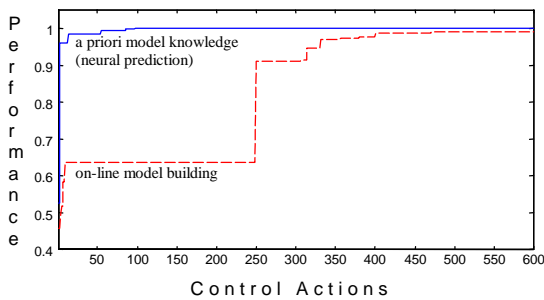


Figure 7 – No *a-priori* model

Once again we note that the process is driven towards the optimal performance, even if we need a higher number of control actions. This is due to the necessity of building a preliminary rough model, filling the performance map with measures of performance. It turns out that in this case *the on-line updating of the performance map suffices to construct a model of the process and to conduct it to the required performance*. Several similar experiments have been done [1] and the results we obtained confirm the above statement, which is the starting point of the next class of experiments.

## 6.2 Adaptivity to non-stationary effects of unknown disturbance evolution

Just to take a step in the direction of a real application, we made two hypotheses:

- **we only know and act on a finite number of the control parameters of the dynamical system;** in the case of the Chua’s circuit we supposed to operate only on the values of  $\beta$ ,  $a$ , and  $b$ , considering  $a$  as a time-varying noise, whose effect produces unpredictable changes in the performance landscape;
- **we have no model for the dynamical process to optimise;** this hypothesis is a consequence of the above-presented first class of experiments (fig. 7).

Hence the *performance model* is just the performance map, which, owing to our assumptions, is a three-dimensional time-varying table.

We have therefore a **dynamic performance landscape**, and the optimal control set has to be dynamical as well. The strength of the approach we propose is in its adaptivity. The control system we introduce, indeed, is able to exploit the reactivity to changes of a “living” system (the *alife* environment), to keep the process on high value of performance, driving it to recover an optimal operating condition whenever a drop occur.

We checked this behaviour by arranging a random discrete variation in time on  $\alpha$  (fig. 8). The corresponding evolution of the process performance, without any kind of control, would be the one dashed in fig. 9. As we can see, even if we start from an optimal situation, the first variation of the unknown parameter throws the performance down to an unacceptable efficiency. With the introduction of the evolutionary control, the performance of the system follows the continuous line, over traced in fig. 9. The improvement of the process efficiency is very remarkable.

The performance is once again subject to drops due to the unknown disturbance, but the control system is effective in recovering the optimal value.

In this experimentation the average performance for the uncontrolled situation was 0.503 (worst case) while in the controlled case it was 0.842 achieving a 34% improvement on the average performance.

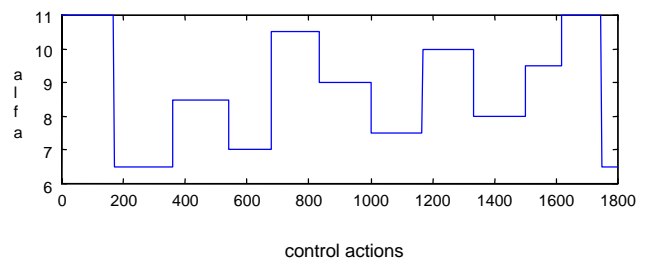


Figure 8 – Random Discrete variations on  $\alpha$  parameter

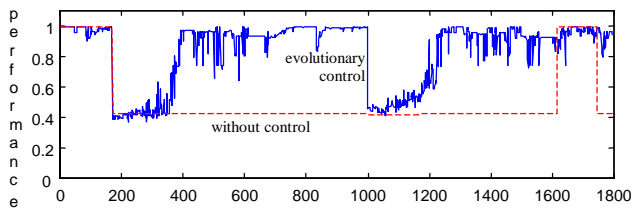


Figure 9 – The Evolutionary Control

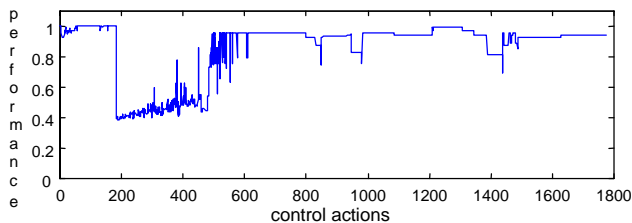


Figure 10 – Infinite Lifetime

This first result surely improves the process performance, continuously pushing it to the optimum, but one could expect on the jaggedness of the fitness curve. In a real system, in fact, this would probably bring to unstableness situation, placing the whole dynamical system in a critical condition.

We argued this was due to the finiteness of each agent's life. The best individual too, in fact, can die, forcing the optimisation scheme to restart searching for a new best value. In the implemented environment each individual has two death causes: aging and fight. Hence we arranged another experiment in which we considered an infinite lifetime for every individual. The resulting graph is showed in fig. 10. In this case the environment will keep on following the process evolution because the artificial environment will continue to update, but the best individual will prevail in every fight owing to its high value of performance. We see a remarkable increase in stability of the performance of the system.

However we also to notice that, in this latter case, on the occasion of the first performance drop, we need a higher number of control actions to recover the optimal condition. This is due to the slower reactivity of the artificial environment to sudden changes in the performance landscape. This is the reason why we have to choose a compromise between the reactivity of the system and its required stability.

## 7. Concluding remarks

We introduced a smart adaptive technique for control and optimisation of complex processes. We based our proposal on the development of an artificial environment evolving in parallel with the process. Exploiting its characteristics of evolution, biodiversity and adaptivity, we succeeded in achieving an on-line

optimisation of the process via a continuous learning and updating of its model.

We tested the proposed methodology on a very well known and widely studied dynamical system: the Chua's circuit. We defined a performance to maximise. Experimentation concerned the on-line optimisation of Chua's circuit in different regimes. We supposed not to know the equations describing the circuit, working on the rough signal generated by a simulator of the circuit. To test the adaptive capability of the alife environment we considered one parameter affected by unknown changes and then we let the alife environment to try to adapt itself to the new condition. In our experimentation we achieved a 34% improvement of the average performance, with respect to the uncontrolled situation. Moreover we saw the system doesn't necessarily need an a priori performance model because it develops a *growing into* knowledge model through continuous learning. The results we showed seem to be very promising for future application on a real process.

## Bibliography

- [1] Annunziato, M., 1999, *Emerging Structure in Artificial Societies*, in Creat. Appl. Lab, SIGGRAPH, LA/CA, USA
- [2] Annunziato, M., Bertini, I., Lucchetti, M., Pannicelli, A., Pizzuti, S., 2001, *Adaptivity of Artificial Life Environment for On-Line Optimization of Evolving Dynamical Systems*, in Proc. EUNITE01, Tenerife, Spain
- [3] Annunziato, M., Bertini, I., Pannicelli, A., Pizzuti, S., Tsimring, L., 2000, *Complexity and Control of Combustion Processes in Industry*, in Proc. CCSI2000, Warwick, UK
- [4] Bedau, M. A., 1992, *Philosophical Aspects of Artificial Life*, in *Towards a Practice of Autonomous Systems*, MIT Press, Cambridge/MA, USA, pp. 494-503
- [5] Chua, L. O., 1992, *The Genesis of Chua's Circuit*, AEU 46, p. 250
- [6] Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge/MA, USA
- [7] Langton, C., 1989, *Artificial Life*, Addison-Wesley, Redwood City/CA, USA, pp. 1-47
- [8] Madan, R. N., 1993, *Chua's Circuit: A Paradigm for Chaos*, World Science Series In Nonlinear Sciences
- [9] Michalewicz, Z., 1992, *Genetic Algorithms + Data Structures = Evolution Programs* Springer-Verlag, Berlin.
- [10] Trojanowski, K., Michalewicz Z., 1999, *Evolutionary Algorithms for Non-Stationary Environments*, Proc. 8th Workshop on Intelligent Information Systems, Ustron, Poland, ICS PAS Press