Chaotic Sequences to Improve the Performance of Evolutionary Algorithms

Riccardo Caponetto, *Member, IEEE*, Luigi Fortuna, *Fellow, IEEE*, Stefano Fazzino, and Maria Gabriella Xibilia, *Member, IEEE*

***Abstract*—This paper proposes an experimental analysis on the convergence of evolutionary algorithms (EAs). The effect of introducing chaotic sequences instead of random ones during all the phases of the evolution process is investigated. The approach is based on the substitution of the random number generator (RNG) with chaotic sequences. Several numerical examples are reported in order to compare the performance of the EA using random and chaotic generators as regards to both the results and the convergence speed. The results obtained show that some chaotic sequences are always able to increase the value of some measured algorithm-performance indexes with respect to random sequences. Moreover, it is shown that EAs can be extremely sensitive to different RNGs. Some t-tests were performed to confirm the improvements introduced by the proposed strategy.**

***Index Terms*—Chaotic sequences, evolutionary algorithm (EA), performance evaluation, random number generators (RNGs).**

I. Introduction

THE CONVERGENCE properties of an evolutionary algorithm (EA) [1]–[8], are strongly connected to the random sequence applied on variation operators during a run. In particular, it can be shown that when different random sequences are used during the evolution, the final results may effectively be very close but not equal. Different numbers of generations may also be required to reach the same optimal values. However, there are no analytical results that guarantee an improvement of the performance indexes of EAs depending on the choice of a particular generator.

Recently, chaotic sequences have been adopted instead of random ones and very interesting results have been shown in many applications such as secure transmission [9], natural phenomena modeling [10], neural networks [11]–[13], and nonlinear circuits [14]. Also in [16], chaotic time series were used in DNA computing procedures. The choice of chaotic sequences is justified theoretically by their unpredictability, i.e., by their spread-spectrum characteristic.

With regard to EAs, in [15], a special mutation operator, applied during gene recombination and based on the chaotic logistic function, was introduced and showed interesting results in enhancing EA exploitation capabilities. More recently in [17], chaotic sequences were used to increase population size dynamically in order to avoid premature convergence. In this paper, chaotic sequences have been used, as specified in Section V, in all the phases of EA, i.e., in the generation of the initial population and during the execution of the selection, crossover, and mutation operators.

TABLE I
EA Parameters Used During Comparison Tests for De Jong Functions

Evaluation number	400
Population size	30
Number of subpopulation	10
Mutation probability	0.001
Crossover probability	0.9
Convergence percentage	0.99
Replacement percentage	0.25
Replacement number	5

By using several experimental optimization tests based on the De Jong functions [2], the linear matrix inequality (LMI) eigenvalue problems [18], the traveling salesman problem (TSP) [2], and the iterated prisoner's dilemma (IPD) [26], it is shown that convergence can be enhanced using particular chaotic series applied to all variation operators. In particular, a comparison between some standard random number generators (RNGs) and chaotic ones is made starting from identical initial conditions, showing the existence of particular chaotic sequences that are able to increase the algorithm-exploitation capability. Moreover, a comparison between the performance obtained by using different seeds in generating the random sequences has been performed. Some t-tests have been performed at each generation comparing different data sets related to the five De Jong functions. These tests were obtained after running the EAs with 50 different initial populations in the cases of random generators and chaotic time series. A different set of t-tests was conducted at each generation for a set of 50 different TS problems in order to confirm the fast convergence of the EAs based on chaotic sequences.

The paper is structured as follows. Section II describes an introduction on RNGs and the three RNGs used during the tests. Section III offers a short introduction on chaotic dynamics and describes the systems generating the chaotic time series used for the tests. Section IV introduces the proposed approach. Section V describes the considered test problems and Section VI reports the results of the tests and offers conclusions.

Manuscript received March 15, 2001; revised October 16, 2001, June 11, 2002, and January 6, 2003. This work was supported in part by the Italian "Ministero dell'Istruzione, dell'Universitá e della Ricerca" (MIUR) under the FIRB project RBNE01CW3M_001.

R. Caponetto, L. Fortuna, and S. Fazzino are with the System and Control Group, DIEES, Facolta' di Ingegneria, Universita' degli Studi di Catania, 95125 Catania, Italy (e-mail: riccardo.caponetto@dees.unict.it; lfortuna@dees.unict.it).

M. G. Xibilia is with the Dipartimento di Matematica, University of Messina, Italy (e-mail: mxibilia@ingegneria.unime.it).

Digital Object Identifier 10.1109/TEVC.2003.810069

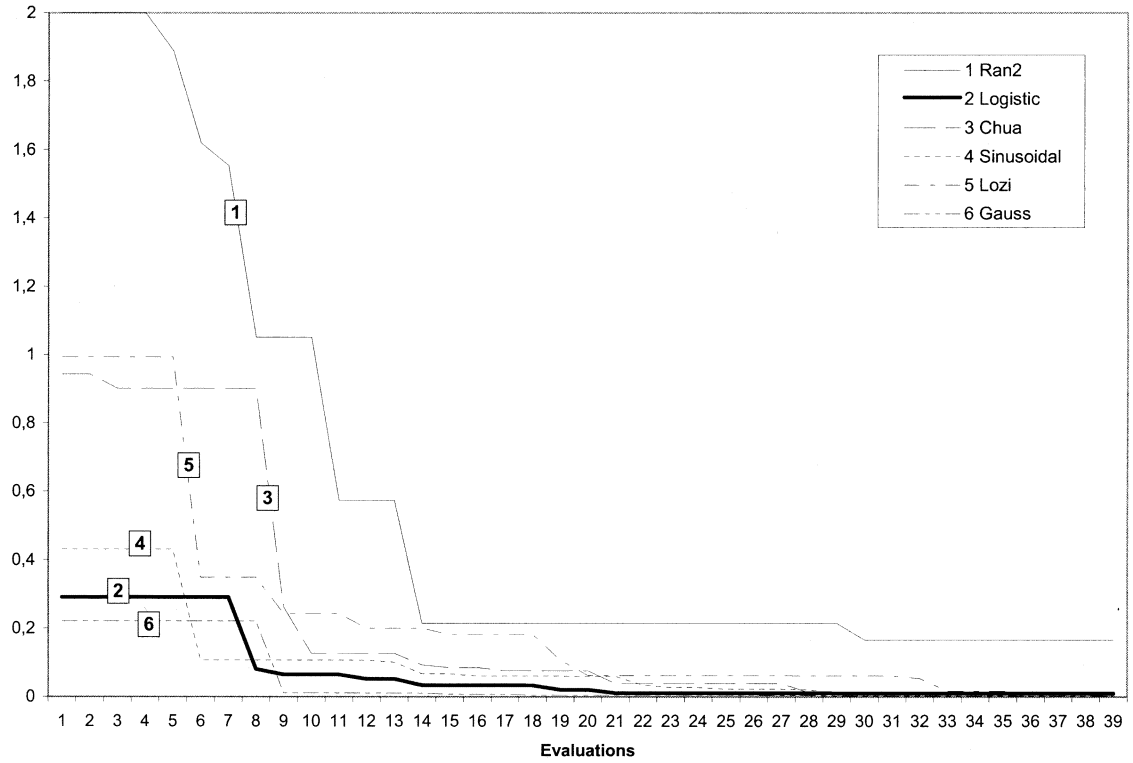

Fig. 1. Best element of population versus the number of evaluations with the different generators for function $f1$.

II. Random Number Generators (RNGs)

The linear congruent generators are common RNGs implemented in ANSI-C libraries. They generate a sequence of integers $I_1, I_2, I_3, \ldots$ between zero and $m-1$ (often called RANDMAX) using the following relation:

$$I_{j+1} = aI_j + c \pmod m \tag{1}$$

where a and c are positive integers and m is the modulo.

The generated sequence is characterized by the values of a, c, m, and by the seed I_0. This series will produce values that will be repeated with a period no longer than m, starting from the seed I_0. From a practical perspective, this generator is very easy to implement and needs a short central processing unit (CPU) time, but some drawbacks can be encountered [19].

Since the ANSI standard imposes that *Rand* function returns a values of the integer type (usually a two-byte quantity), the maximum RANDMAX value is often not so large. Another drawback of the linear congruent generators is that they are not free from sequential correlation on successive calls. In fact, if k random numbers at a time are used to plot points in a k-dimensional space, the points will not tend to cover the k-dimensional space, but rather they will accumulate on $(k-1)$ dimensional planes. A further limitation of these generators is that their low-order bit is often much less random than their high-order bit. Another commonly applied generator, proposed in [20], uses a simple multiplicative congruental algorithm of the form

$$I_{j+1} = aI_j \pmod m \tag{2}$$

that can be as good as any of the more general linear congruental generators that have $c \neq 0$ in equation (1).

This portable RNG, named *minimal standard*, uses the following values for the parameters: $a = 7^5 = 16\,807$, $c = 0$, $m = 2^{31} - 1$.

Equation (2) requires a high-level language to be implemented to overcome the finite length of the variable, but it is possible to solve this problem by adopting an *approximate factorization* of m according to:

$$m = aq + r \tag{3}$$

with

$$q = [m/a], \quad r = m \mod a. \tag{4}$$

It has been proved [19] that if $r < q$ and $0 < z < m-1$ then $a(z \mod q)$ and $r[z/q]$ belong to the interval $0, \ldots, m-1$ and

$$a(z \mod m) = \begin{cases} a(z \mod q) - r[z/q], & \text{if } \geq 0 \\ a(z \mod q) - r[z/q] + m, & \text{otherwise} \end{cases}. \tag{5}$$

Using for a, c, and m the above reported values and $q = 127\,773$ and $r = 2836$, the period of this generator is $2^{31} - 22.1 * 10^9$.

This algorithm represents the core of two of the three RNGs adopted in our study. The first RNG is based on the minimal standard algorithm for the random values, but it shuffles the output to remove low-order serial correlations. The shuffling algorithm is that of Nays and Durhan [21]. Later, this algorithm will be used and labeled Rand1. In order to increase the period of the previous

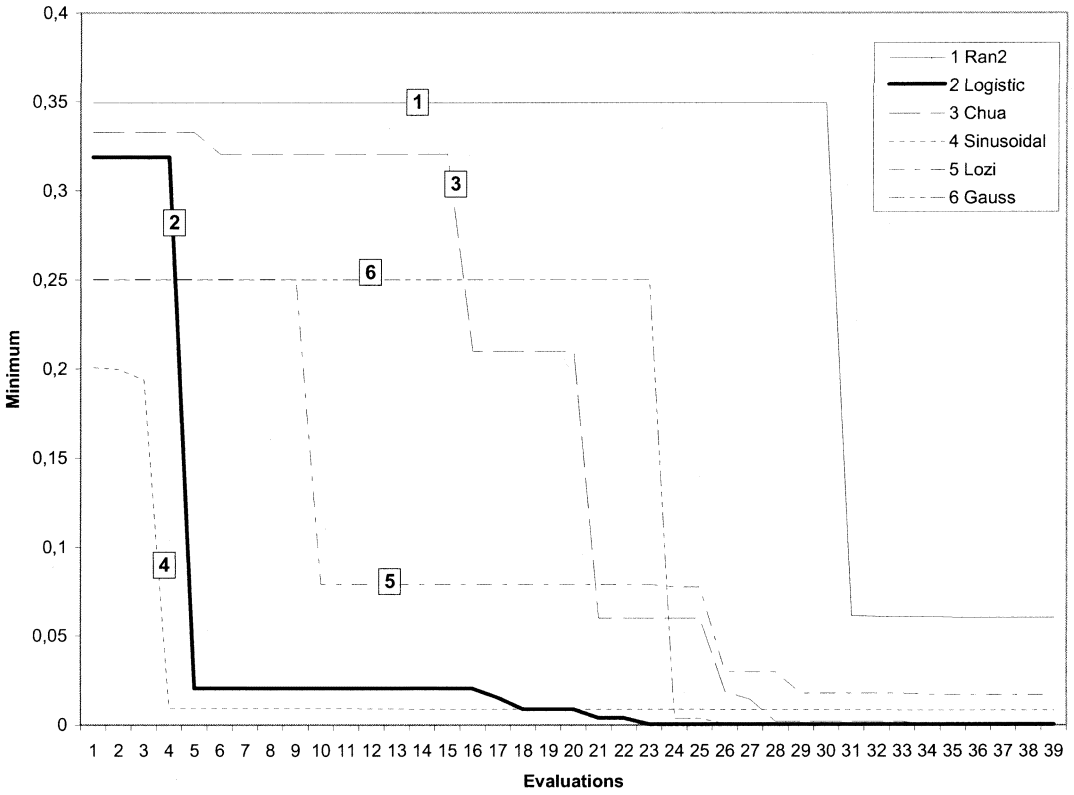

Fig. 2. Best element of population versus the number of evaluations with the different generators for function $f2$.

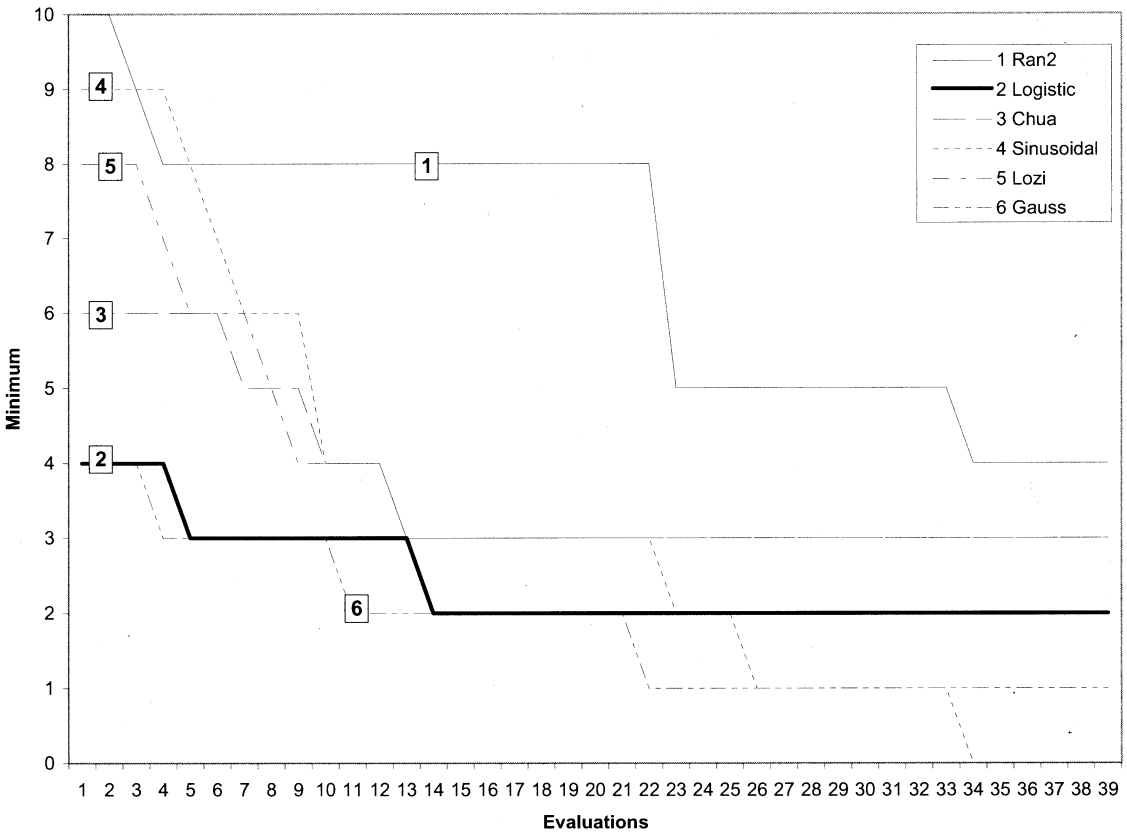

Fig. 3. Best element of population versus the number of evaluations with the different generators for function $f3$.

generator, L'Ecuyer [22] proposed a new algorithm with a period of 10^{18}. This procedure is very useful and popular and it will be labeled $\mathrm{rand2}$. The last RNG used in our work, labeled $\mathrm{Rand3}$, is based on a *subtractive method* as proposed by Knuth [21].

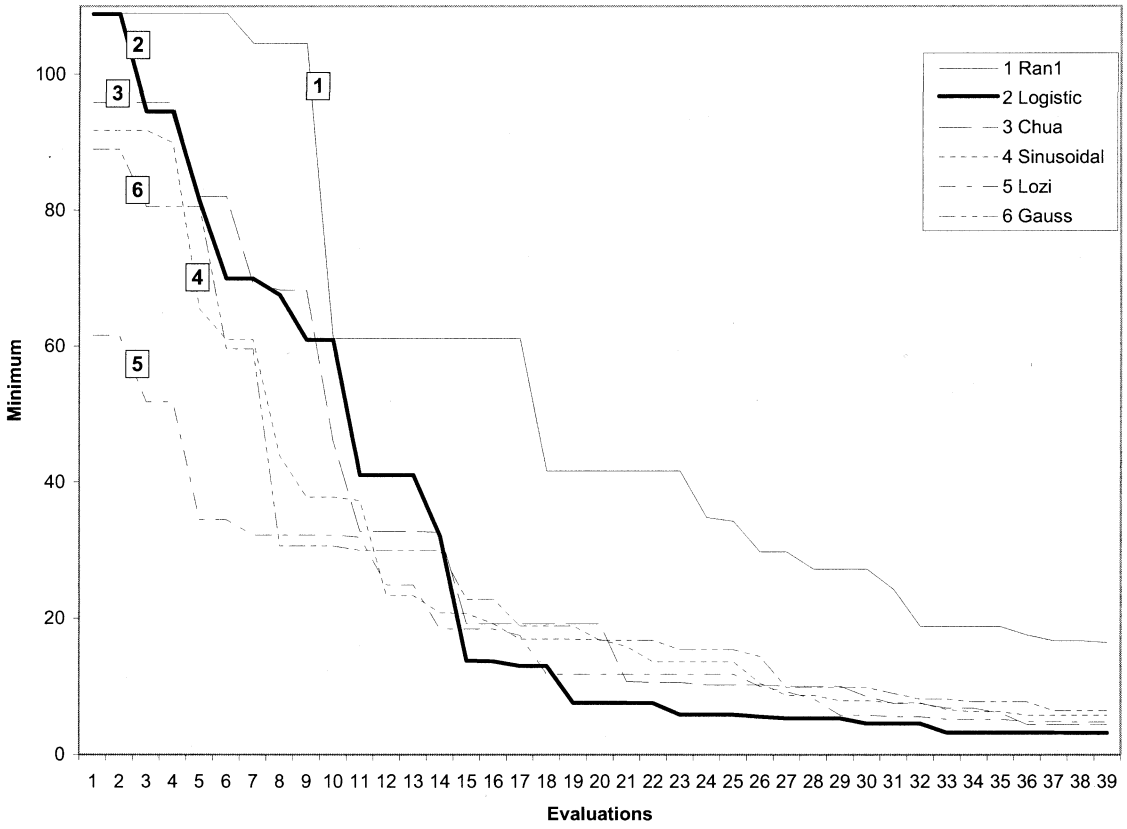

Fig. 4. Best element of population versus the number of evaluations with the different generators for function $f4$.

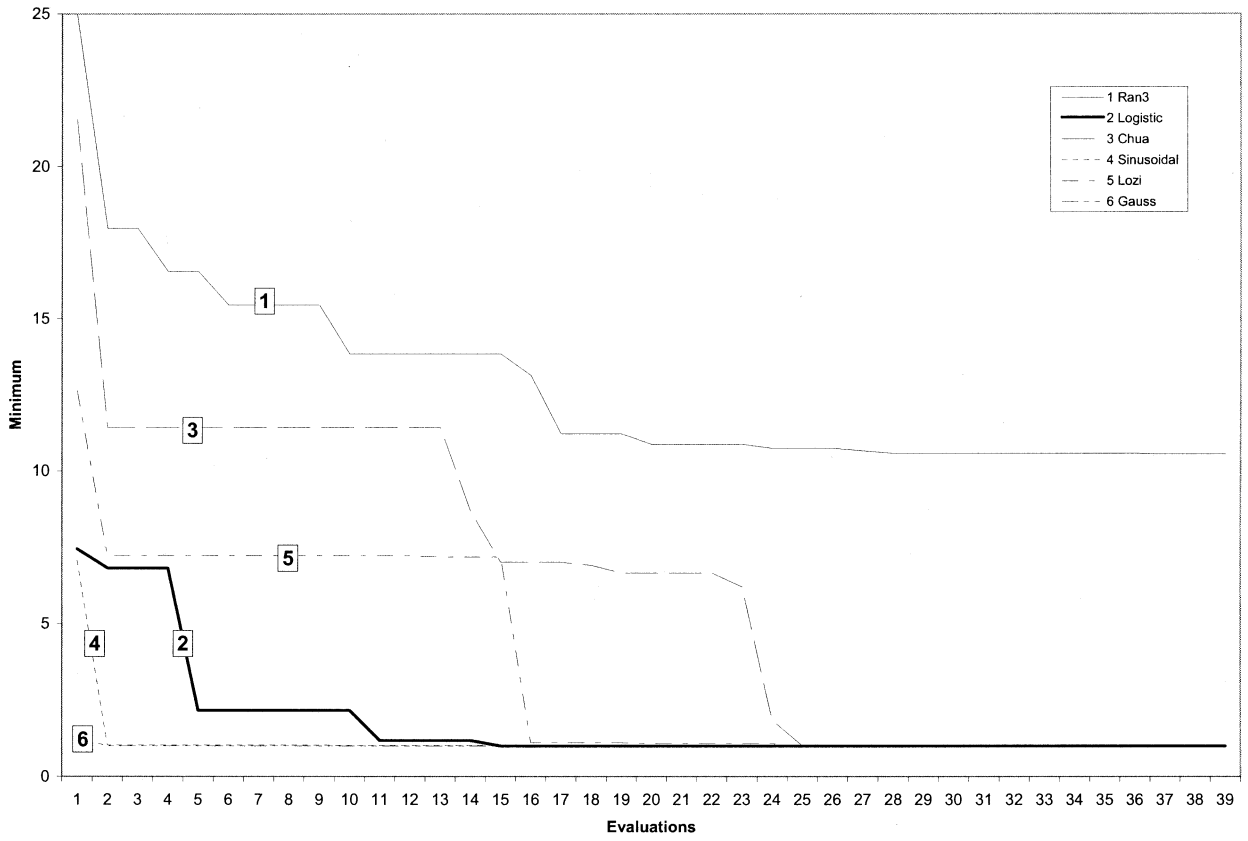

Fig. 5. Best element of population versus the number of evaluations with the different generators for function $f5$.

III. Chaotic Dynamics Based Number Generators

Several characterizations of a chaotic system are possible. Before giving a definition of chaos, the following definitions are required [24].

Definition 1: Given a dynamic system of the form $dx/dt = f(x)$, not satisfying Liouville's Theorem (conservative systems theorem) and that contracts its volume in all (or at least some) parts of phase space, the set of points in the phase space where the volumes are contracted is called an attractor.

Definition 2: An attractor can have an integer dimension (e.g., points have dimension zero, lines have dimension one, plane surfaces have dimension two, etc.), or they have fractal dimension: in this case, they are called a strange attractor. For the case of a map $dx/dt = G(x)$, a volume contraction or expansion is determined by computing the determinant J of Jacobian matrix. Then, all points for which $J < 1$ define the contraction subspace; correspondingly, all points for which $J > 1$ define the expansion subspace and, finally, the subset of the phase space where $J = 1$ indicates a limit cycle.

From a time-series perspective, a chaotic system is characterized by signals with a broad-band spectrum that depend strongly on the initial conditions. A more analytical way to define it makes use of the Lyapunov exponents, which are closely related with the qualitative aspects reported above.

Consider an orbit $x(t)$ whose long-term limit fills out the attractor. Now consider the initial condition for this orbit $x(0)$ and a second initial condition $x'(0)$, which is displaced from $x(0)$ by an infinitesimal distance $\delta x(0)$

$$x'(0) = x(0) + \delta x(0). \tag{6}$$

If there is an orientation of $\delta x(0)$ such that

$$h = \lim_{t\to\infty} \frac{1}{t}\left(\frac{\delta x(t)}{\delta x(0)}\right) > 0 \tag{7}$$

then the attractor is chaotic and the quantity h is called a Lyapunov exponent.

The chaotic time series sequences used in our experiments are now reported.

A. Logistic Map

One of the simplest dynamic systems evidencing chaotic behavior is the iterator named *logistic map* [23], whose equation is the following:

$$x_{k+1} = a x_k (1 - x_k). \tag{8}$$

The following values for the parameters $x_0 = 0.2027$ and $a = 4$ have been used for simulation.

B. Tent Map

The Tent map [24] resemble the logistic map and assumes the following form:

$$x_{k+1} = G(x_k) \tag{9}$$

TABLE II
EA Parameters Used During Comparison Tests for LMI Eigenvalues Problem

Generations number	2000
Population size	200
Number of subpopulation	10
Mutation probability	0.001
Crossover probability	0.9
Convergence percentage	0.99
Replacement percentage	0.25
Replacement number	5

TABLE III
EA Parameters Used During Comparison Tests for IPD Problem

Generations number	50
Population size	20
Mutation probability	0.001
Crossover probability	0.7
Elitism	Yes
Scaling	Sigma-Truncation
Number of tournaments in IPD	10

TABLE IV
EA Parameters Used During Comparison Tests for TS Problem

Generations number	100
Population size	40
Number of subpopulation	5
Mutation probability	0.1
Crossover probability	1
Convergence percentage	0.99
Replacement percentage	1.0
Replacement number	5

with

$$G(x) = \begin{cases} x_k/0.7, & \text{if } x < 0.7 \\ \frac{1}{0.3}\, x_k(1 - x_k), & \text{otherwise} \end{cases}. \tag{10}$$

The initial condition of x has been fixed at $x_0 = 0.27$.

C. Sinusoidal Iterator

A third chaotic generator used in this paper is the so-called *sinusoidal iterator* [24] represented by

$$x_{k+1} = a\, {x_k}^2 \sin(\pi x_k). \tag{11}$$

In what follows, it is iterated with $a = 2.3$ and $x_0 = 0.7$ and simplified by using the following relation:

$$x_{k+1} = \sin(\pi x_k). \tag{12}$$

D. Gauss Map

This transformation is similar to a quadratic one and is adopted in the literature [24] for testing purposes because it allows a complete analysis of its chaotic qualitative and quantitative features. The equations are

$$x_{k+1} = G(x_k) \tag{13}$$

$$G(x) = \begin{cases} 0, & \text{if } x = 0 \\ \dfrac{1}{x} \bmod 1, & x \in (0,\, 1) \end{cases}. \tag{14}$$

TABLE V
PERFORMANCE WITH DIFFERENT SEQUENCES—FUNCTION $f1$

f1	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossovers	10000	8999	10000	9393	9288	9478	9285	9285
Mutations	4200	4732	4611	1741	6811	27009	6865	6865
Genome eval.	9911	9383	9831	9479	9666	9926	9666	9686
Max Score	37.554749	34.118866	34.856255	33.735725	38.918758	34.499832	33.437473	36.93539
Min Score	0	0	0	0	0	0	0	0
On-line	0.4619659	0.301510	0.610671	0.150419	0.275765	0.178527	0.229193	0.192974
Off-line max	1.733913	0.490048	0.731344	0.30012	0.48838	0.371799	0.413135	0.387479
Off-line min	0.4567389	0.056951	0.551391	0.007396	0.022458	0.005274	0.035566	**0.004825**

E. Lozi Map

Lozi's piecewise linear model [24] is a simplified version of Henon's attractor and it admits a strange attractor. The transformation is given by

$$(x_{k+1}, y_{k+1}) = H(x_k, y_k) \tag{15}$$

with

$$H(x_k, y_k) = (1 + y_k - a \mid x_k \mid, bx_k). \tag{16}$$

Lozi suggested the values $a = 1.7$ and $b = 0.5$ for the parameters.

F. Chua's Oscillator

Chua's oscillator [23] has been studied extensively because of its extremely rich variety of dynamical behaviors together with a relatively simple mathematical model. Its dimensionless state equations are

$$\begin{aligned} \dot{x(t)} &= \alpha(y(t) - l(x(t))) \\ \dot{y(t)} &= x(t) - y(t) + z(t) \\ \dot{z(t)} &= -\beta y(t) - \gamma z(t) \end{aligned} \tag{17}$$

with

$$l(x) = m_1 x + 0.5(m_0 - m_1)(\mid x + 1 \mid - \mid x - 1 \mid) \tag{18}$$

x, y, and z being the state variables and α, β, γ, m_0, and m_1 the five dimensionless system parameters.

The *double scroll* attractor is observed in Chua's oscillator if

$$\alpha = 9, \quad \beta = 14.286, \quad \gamma = 0, \quad m_0 = -1/7, \quad m_1 = 2/7. \tag{19}$$

Discrete-time chaotic time series are derived by using the Chua oscillator signals at suitable sampling times.

IV. PROPOSED APPROACH

As outlined in the introduction, some applications of chaotic systems in EAs and in optimization problems have been presented [15], [17]. In particular, in [15], the use of the chaotic logistic function for the definition of a special mutation operator has been discussed, and some results have been presented showing how this *chaotic* mutation operator aids exploration in the search space. The IPD has been used as an example.

In this paper, sequences generated from chaotic systems substitute random numbers in all phases of EAs where it is necessary to make a random-based choice. The only difference with the classical EA, implemented in this paper by using GaLib 2.4.2, is the source of the random numbers, which are not generated by RNGs, but by iterating of one step the chaotic map, i.e., each time a random number is needed by the classical EA it is generated by iterating one step of the chosen chaotic

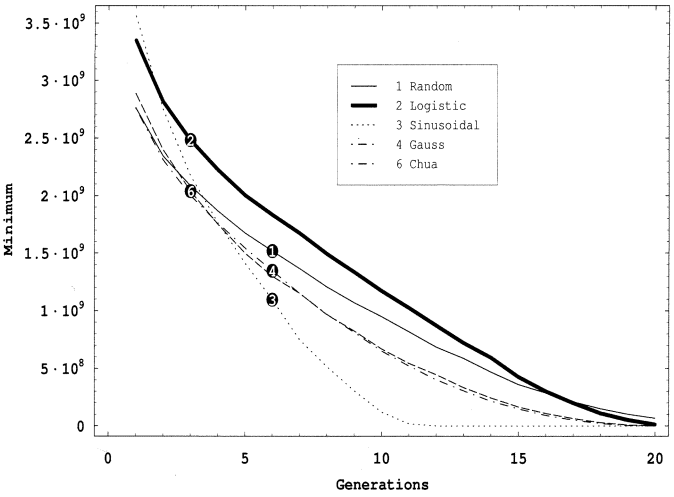

Fig. 6. Best element of population versus the number of generations with the different generators for LMI test.

TABLE VI
OFFLINE PERFORMANCE FOR RNGS—FUNCTION $f1$

f1	*Rand 1*	*Rand 2*	*Rand 3*
Seed 1	3.453755	0.48123	3.03849
Seed 2	3.341274	2.82592	3.490818
Seed 3	1.606754	4.881926	2.189255
Seed 4	1.676377	0.619409	**0.551391**
Seed 5	**0.4567389**	0.400295	0.916706
Seed 6	0.644772	0.684008	2.0624
Seed 7	3.271875	**0.056951**	4.519243
Seed 8	5.517787	4.567389	1.746226
Seed 9	1.656444	3.274331	1.781544
Seed 10	1.262267	0.551391	5.158041
Seed 11	0.513702	1.765736	4.567389
Seed 12	3.436308	0.495583	4.567389

TABLE VII
PERFORMANCE WITH DIFFERENT SEQUENCES—FUNCTION $f2$

f2	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossover	8962	9123	9661	8711	8911	7199	9201	8109
Mutation	3153	4001	3355	3312	3378	3001	3099	3241
Genome eval.	9256	9311	9356	9321	9101	9351	9255	9256
Max Score	550.9271	891.3720	505.3175	595.6082	842.8812	805.6378	826.1843	586.5429
Min Score	0	0	0	0	0	0	0	0
On-line	22.254213	2.116764	3.297343	1.22038	1.14481	0.977754	1.343404	1.011361
Off-line max	34.798775	5.581415	4.797675	3.761769	4.120534	3.888743	4.575599	3.299497
Off-line min	1.437763	0.192914	1.671401	**0.004418**	0.015798	0.009942	0.010464	0.005295

map that has been started from a random initial condition at the first iteration of the EA. In particular, the use of chaotic sequences affects the following EA phases.

1) During the creation of the initial population, the chaotic sequences are used to generate the individuals.
2) During the selection algorithm, the chaotic sequences are used for the probabilistic choice of individuals according to the roulette wheel method [2].
3) During the crossover algorithm, the chaotic sequences are used for the choice of points inside the chromosomes or for the generation of bit masks and to decide whether or not to apply the operator.
4) During the mutation algorithm, the chaotic sequences are used for choosing the positions of the chromosome's bits to be changed and to decide whether or not to apply the operator.

Therefore, chaotic sequences influence the behavior of all operators (mutation, crossover), not because new operators are introduced (as in [15]), but because all the existing standard operators work following the outcomes of a chaotic sequence instead of a standard random generator.

In order to compare the behavior obtained with random and chaotic sequences, an EA that is based on the shareware software GaLib 2.4.2 and a *chaotic engine*, written in C, which implements all the chaotic generators, have been used. According to the structure of the GaLib 2.4.2 software, routines can be changed or extended for the generation of all random numbers used in the operators of EAs. Then, the *chaotic engine* extends the basic routines for generating random numbers by introducing new C functions implementing the equations of various chaotic systems.

The GaLib software includes different evolutionary algorithm, in particular the basic and the steady-state algorithms, and it has been chosen because it implements the RNG routines as a module that can be extended easily by the chaotic engine routines. No other changes have been made in the GaLib library, neither in operator routines implementation, nor in the internal representation of chromosomes.

V. TEST PROBLEMS

Different types of test problems have been considered to compare the use of chaotic sequences and RNGs in the EAs. The first group is the well-known De Jong functions, the second is an LMI eigenvalues problem, and the last one is the IPD. For comparison, the same initial population has been used for each type of EA. This is slightly different from the procedure outlined above, in which the initialization should be created by using the random or chaotic sequence generators. Moreover, both the De Jong functions and a set of TS problems have been used to perform a statistical t-test to assess the convergence properties of the proposed strategy.

TABLE VIII
OFFLINE PERFORMANCE FOR RNGS—FUNCTION $f2$

f2	*Rand 1*	*Rand 2*	*Rand 3*
Seed 1	1.51593	0.64792	2.458856
Seed 2	2.779665	0.544831	**1.671401**
Seed 3	1.644584	0.950852	2.135822
Seed 4	1.93931	**0.192914**	3.318807
Seed 5	1.502641	1.663608	1.76978
Seed 6	2.14627	0.242475	3.332593
Seed 7	4.059099	0.753788	1.671401
Seed 8	3.223777	0.435486	3.363044
Seed 9	**1.437763**	1.042497	4.227733
Seed 10	2.134544	0.249408	4.21593
Seed 11	1.718639	2.128659	1.779665
Seed 12	3.08363	0.355303	2.644584

Besides the optimal solutions, some performance indexes have been computed for each experiments, in particular the online and the offline performance [2]. These results are reported in Section VI along with some comments, while in this section only the experimental setups are described.

All the standard number generators have been used starting from fixed seeds having the values: 1, 2, 100, 200, 1000, 2000, 100 000, 200 000, 1 000 000, 2 000 000 (in the reported Tables, these conditions are reported as $\mathrm{Seed1}$ to $\mathrm{Seed12}$), while for the chaotic systems the adopted parameters are the same as those introduced in the previous section.

The number of crossovers, the number of mutations, the number of genome evaluations, the maximum and minimum score, online performance and offline min and max performance have been monitored for each experiment.

A. De Jong Functions

During the De Jong functions [2] optimization test, EA parameters were set as in Table I. The five functions taken into consideration, which were minimized, are

$$f_1(x) = \sum_{i=0}^{n} x_i^2; \qquad \text{with } -5.12 \le x_i \le 5.12 \tag{20}$$

$$f_2(x) = 100(x_0^2 - x_1)^2 + (1 - x_0)^2; \qquad \text{with } -2.048 \le x_i \le 2.048 \tag{21}$$

TABLE IX
PERFORMANCE WITH DIFFERENT SEQUENCES—FUNCTION $f3$

f3	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossover	4101	4308	4413	4521	4674	4711	4312	4419
Mutation	259	371	367	376	365	389	371	313
Genome eval.	4432	4356	4511	4356	4789	4677	5211	4982
Max Score	25	26	27	24	26	25	24	24
Min Score	0	0	0	0	0	0	0	0
On line	5.039601	3.930898	4.344199	2.22785	3.2644349	0.51285	3.235349	1.3111
Off-line max	5.16499	4.064998	4.5	2.355001	3.382499	0.6725	3.347502	1.455
Off-line min	4.867497	3.69999	4.139998	2.0425	3.087498	**0.3125**	3.08499	1.085

$$f_3(x) = 30 + \sum_{i=1}^{5} |x_i|; \qquad \text{with } -5.12 \le x_i \le 5.12 \tag{22}$$

$$f_4(x) = \sum_{i=1}^{30} i x_i^4 + \text{Gauss}(0, 1); \qquad \text{with } -1.28 \le x_i \le 1.28 \tag{23}$$

$$\frac{1}{f_5(x)} = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}; \qquad \text{with } -65.536 \le x_i \le 65.536. \tag{24}$$

Steady-state EAs, binary strings, elitism, and single-point crossover were adopted.

In our implementation, mutation was applied with low probability (0.001) and modified elements in chromosomes and, as usually, provides a guarantee to recover good solutions that may be lost through the action of selection and crossover.

Tables V to XIV show for each of the five functions the results obtained with the five chaotic systems and with the RNG (*Rand 1*, *Rand 2*, and *Rand 3*).

Figs. 1–5 show, again for each function, the trend of the best element of population for each evaluation. Only the trend at early evaluation has been reported in order to appreciate the fast convergence of the algorithm using chaotic sequences. Comments about tables and figures are reported in Section VI along with the results of the t-tests.

B. LMI Eigenvalues Problem

Another test was performed solving an LMI eigenvalue problem via an EA, again comparing standard RNGs and the proposed chaotic sequences.

A standard LMI problem assumes the following form:

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i > 0 \tag{25}$$

where $x \in R^m$ is the variable to be optimized and $F_i \in R^{n*n}$ are symmetric matrices.

Relation (25) is equivalent to n polynomial inequalities obtained by imposing that all the principal minors of $F(x)$ be positive [18].

TABLE X
OFFLINE PERFORMANCE FOR RNGS—FUNCTION $f3$

f3	*Rand 1*	*Rand 2*	*Rand 3*
Seed 1	**4.867497**	3.812502	4.45
Seed 2	5.6525	4.139999	4.455001
Seed 3	5.220001	4.247501	4.62
Seed 4	7.33	4.340001	**4.139998**
Seed 5	4.939998	**3.69999**	4.5275
Seed 6	5.699999	6.300001	4.664999
Seed 7	5.874999	4.5025	4.135002
Seed 8	4.950001	4.5425	5.077501
Seed 9	5.427501	4.282501	5.912499
Seed 10	5.069999	4.114999	5.590001
Seed 11	5.71	4.257499	4.939998
Seed 12	4.975	4.56	4.950001

In particular the eigenvalues problem consists in minimizing the maximum eigenvalues of a matrix while also respecting LMI constraints. So the general form of the problem is the following:

$$\begin{cases} \min \lambda \\ A(x) - \lambda I < 0 \\ B(x) > 0 \end{cases}$$

or

$$\underset{B(x)>0}{\text{Min}} \ \lambda_{\max}\left(A(x)\right)$$

where $A(x)$ and $B(x)$ are affine symmetric matrices.

The particular LMI eigenvalues problem taken into account is

$$\begin{cases} AP + PA^T + Q < 0 \\ P > 0 \end{cases}$$

where

$$A = \begin{pmatrix} -7 & 0 & -2 & 0 & -3 \\ 0 & -8 & 0 & -1 & -1 \\ -2 & 0 & -9 & -5 & 0 \\ 0 & -1 & -5 & -24 & 0 \\ -3 & -1 & 0 & 0 & -18 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

with $A < 0$ and $Q > 0$.

TABLE XI
PERFORMANCE WITH DIFFERENT SEQUENCES—FUNCTION $f4$

f4	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossovers	2505	2505	2505	2567	2504	2678	2321	2576
Mutations	11108	11200	11208	12311	11208	11299	10200	11200
Genome eval.	2830	2830	2830	2900	2840	2830	2821	2830
Max Score	405.23	444.22998	389.12	501.365051	516.148438	370.25	391.911	425.4707
Min Score	0	0	0	0	0	0	0	0
On-line	8.7286	9.75886	9.15886	5.573584	6.296947	5.406761	6.4243	6.273797
Off-line max	10.1233	11.957835	11.715	7.350011	8.142813	7.081879	8.13566	8.1749
Off-line min	4.1234	5.286104	5.1222	2.804984	3.022	**2.704**	3.41	2.88317

Due to the fact that $P \in R^{(5*5)}$ is symmetric, the considered optimization problem with $n*(n+1)/2 = 15$ variables is equivalent to

$$\begin{cases} \min\ \lambda \\ AP(x) = P(x)A^T - \lambda I < 0 \\ P(x) > 0. \end{cases}$$

Regarding the fitness function, the following relations were used:

$$\begin{cases} 10^8(AP + PA^T + Q), & \text{if } \lambda_{\max} \geq 0 \\ 2 * \lambda_{\max}(AP + PA^T + Q) + 4 * STD(\lambda), & \text{otherwise} \end{cases} \tag{26}$$

where STD is the standard deviation. This choice allows us to minimize the maximum eigenvalue of the matrix $AP+PA^T+Q$ by also imposing that it be negative.

To face the LMI eigenvalues problem with a standard optimization approach, the MatLab LMI toolbox, based on the interior point algorithm has been also used. During optimization, the EAs were used setting the parameters as reported in Table II.

Tables XV and XVI report the maximum values of the eigenvalues problem obtained during the test by using all the random and chaotic generators, together with the reports on the monitored quantities. Fig. 6 reports the trend of the best element of the population versus the number of generation.

C. Iterated Prisoner's Dilemma (IPD)

As a further investigation of the proposed conjecture, an IPD was addressed. The IPD is a well-known two-agent iterated game defined by a payoff matrix where each player has two options called *cooperation* and *defection*. The objective of the game is for each agent to maximize its score. While in the single-play version the only rational move is defection, in the iterated version the score of multiple plays is summed, making the choice of an optimal strategy not straightforward. EAs are well suited to face this problem [25]. According to the implementation presented in [25] and [26] an IPD strategy depends on the history of three previous moves, together with six initial moves. Since there are two possible moves for each player, each strategy is represented by a binary string of 70 bits and constitutes an individual. The fitness of each individual is evaluated by playing its strategy against all the individuals of the population for ten iterations and computing the total average score per opponent. We chose the parameters for the EAs according to the values given in [15] and reported in Table III.

TABLE XII
OFFLINE PERFORMANCE FOR RNGS—FUNCTION $f4$

f4	*Rand 1*	*Rand 2*	*Rand 3*
Seed 1	5.10942	6.896307	5.816621
Seed 2	5.203821	4.822581	4.614461
Seed 3	7.491859	8.497499	4.574267
Seed 4	5.622344	**5.286104**	8.255692
Seed 5	5.612545	7.57919	**5.1222**
Seed 6	12.3641	6.200651	6.139767
Seed 7	5.414807	6.501554	5.15505
Seed 8	4.828314	4.511461	6.862654
Seed 9	3.999106	6.711645	8.024367
Seed 10	8.340029	4.68033	6.232791
Seed 11	**4.1234**	4.610773	3.079414
Seed 12	5.816621	6.042168	6.244559

Table XVII reports the obtained results. Also in this case we compared the performance obtained by EAs using chaotic generators versus those using standard RNGs. In particular the logistic, sinusoidal, Gauss, Tent, and Lozi chaotic maps were taken into account, while the three standard algorithms *Rand 1*, *Rand 2*, and *Rand 3* [19] were considered for the standard RNGs.

D. Traveling Salesman Problem (TSP)

The TSP is one of the best-known yet difficult to solve optimization problems. A salesman must visit n cities, passing through each city only once, beginning from one of them, which is considered as the starting base, and returning to it at the end of his tour.

The cost of transportation among the cities (whichever combination possible) is given. The program of the journey is requested, that is, the order of visiting the cities in such a way that the total cost of travel is minimized.

Label the cities from 1 to n, and let city 1 be the starting city of the salesman. Assume that $C(i, j)$ is the visiting cost from city i to city j. There can be $C(i, j) <> C(j, i)$. Apparently, all the possible solutions are $(n-1)!$. It is possible to try to find, exhaustively, all of them, i.e., find the cost for each and every one of these solutions, and finally keep the one with minimum cost. This requires at least $(n-1)!$ steps.

We consider a set of 50 TSPs, each with $n = 20$ cities and different distances between the cities in order to perform a sta-

TABLE XIII
PERFORMANCE WITH DIFFERENT SEQUENCES—FUNCTION $f5$

f5	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossovers	9122	9004	9911	9911	9310	9823	9103	9010
Mutations	3187	3193	3187	3187	3101	3422	3622	3154
Genome eval.	9388	9388	9381	9422	9371	9375	9784	9390
Max. Score	249.999	249.999	249.999	249.999	249.999	249.999	249.99	249.99
Min. Score	1	1	1	1	1	1	1	1
On-Line	9.111	12.8496	8.222	2.7393	3.302981	2.952256	3.178058	2.825897
Off-Line Max	11.223	14.6525	12.222	4.239549	4.941926	4.856848	4.715135	4.558854
Off-Line Min	8.222	10.722871	7.47052	1.059035	1.461008	0.997173	1.215786	**0.996024**

tistical t-test as described in Section VI. The EA parameters as reported in Table IV. The standard crossover operator was modified, adopting the so-called edge recombination crossover as described in [3].

VI. RESULTS ANALYSIS

The following consideration can be made taking into account Tables V–XIV and Figs. 1–5.

Regarding the first De Jong function, denoted with $f1$, all sequences brought the algorithm to the optimal solution. As reported in the Table V, the lowest value for the offline performance was achieved using the Gauss map. Moreover, Fig. 1 shows the trend of the best element of population at each generation for each chaotic generator and for the best random generator. The figure shows that chaotic generators offered the best performance in each case. In Table VI, the performance (offline min) for each random algorithm is reported for 12 different seed values. The table shows that performance was strongly dependent both on the type of random sequence and on the seed.

The same considerations can be made for function $f5$ as shown in Table XIII. Again, the offline performance was better if the chaotic Gauss map was used. As can be noted in Table XIV, also in this case the EA performance was strongly dependent on the adopted RNG. The trends of the best element of the population at each generation can be observed in Fig. 5.

For function $f2$, the previous considerations can be made again, even if the best offline value was achieved using the logistic map, as reported in Tables VII and VIII and Fig. 2. Similar considerations can be made for functions $f3$ and $f4$.

The example concerning the LMI problem confirms the capability of the chaotic sequences to decrease convergence time, as observed in Fig. 6. As reported in Tables XV and XVI, the use of *logistic* and *Gauss* maps yielded the best performance.

Concerning the IPD problem, Table XVII shows the results obtained by running the EAs for 100 times for each considered generator. In particular, the mean offline performance, the standard deviation, and the maximum value reached were recorded. As shown in Table XVII, also in this problem, there was at least one chaotic generator, in particular the logistic map, that performed better then all the considered standard RNGs.

The obtained results essentially show that chaotic sequences enhanced the exploitation capability of the evolutionary search in each of the considered problem, both in term of performance and convergence speed.

To investigate if the results obtained depended on the choice of the initial population and to verify the fast convergence of

TABLE XIV
OFFLINE PERFORMANCE FOR RNGS—FUNCTION $f5$

f5	*Rand 1*	*Rand 2*	*Rand 3*
Seed 1	9.32882	10.970213	9.005498
Seed 2	8.997173	11.279284	9.024902
Seed 3	9.0781	11.009631	10.032846
Seed 4	9.082791	**10.722871**	9.209256
Seed 5	9.993018	12.118708	9.070819
Seed 6	**8.222**	11.271693	9.029374
Seed 7	9.150279	12.005805	9.255081
Seed 8	9.365177	13.042761	9.30967
Seed 9	9.338694	13.133023	9.254108
Seed 10	9.009767	12.436664	**7.47052**
Seed 11	9.38375	11.560422	9.664461
Seed 12	9.215185	12.425977	9.4785

the chaotic EAs, statistical t-tests on the De Jong functions were performed at each generation. The t-tests were used to assess whether the means of two groups were different from each other.

In particular, considering two normal populations X and Y with unknown means μ_X and μ_Y and unknown variances σ_X^2 and σ_Y^2, the t-test, computed on two random samples of sizes n_X and n_Y taken from populations X and Y respectively, is used to test the hypothesis that the two means are equal [28]. In this case, for each t-test, X is the sequence of the mean fitness computed on the entire population at each generation for the EAs running with different initial populations and using the best standard random generator. In particular, $n_X = 50$ different samples were considered. Y is the sequence of the mean fitness computed on the entire population at each generation for the EAs running with different initial populations and using a chaotic generator. Also, in this case, 50 samples were used. The t-tests were applied as follows. The following quantities were computed [28]:

$$S_X^2 = \frac{\sum_{i=1}^{i=n_X} (x_i - \overline{x}\,)^2}{n_X - 1} \tag{27}$$

where x_i is the ith element of the sequence X and $\overline{x}$ is the mean value of the sequence X computed on the 50 samples, and

$$S_Y^2 = \frac{\sum_{i=1}^{i=n_Y} (y_i - \overline{y}\,)^2}{n_Y - 1} \tag{28}$$

where y_i is the ith element of the sequence Y and $\overline{y}$ is the mean value of the sequence Y computed on the 50 samples.

TABLE XV
EIGENVALUES OBTAINED WITH DIFFERENT SEQUENCES—LMI PROBLEM

Rand 1	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Gauss*	*Interior point*
-6.904	-1.392	-8.314	**-8.48**	-4.244	-8.019	**-9.059**	-1.0239

TABLE XVI
PERFORMANCE WITH DIFFERENT SEQUENCES—LMI PROBLEM

LMI problem	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Chua*	*Sinusoid.*	*Lozi*	*Gauss*
Crossovers	8919	9016	8966	8339	8840	8395	9310	9231
Mutations	142	131	181	3455	224	4128	271	298
Genome eval.	9132	9231	9184	9088	9067	9136	9535	9451
Max score	5.14993 e+009	5.53324 e+009	6.29229 e+009	7.44192 e+009	5.84497 e+009	1.03862 e+0010	4.43962 e+009	6.30002 e+009
Min score	45.4279	90.091	18.7271	49.6705	26.6975	16.1943	46.1829	25.6538
On line	8.61992 e+007	9.50971 e+007	8.06292 e+007	1.13837 e+008	5.49004 e+007	6.21775 e+007	1.59525 e+008	7.87812 e+007
Off-line max	1.31158 e+007	1.40886 e+007	1.25346 e+007	1.71860 e+008	9.97999 e+007	1.20831 e+007	2.05056 e+008	1.21592 e+008
Off-line min	1.56451 e+007	1.51545 e+007	1.54388 e+007	3.36554 e+007	391195	172665	6.314 e+008	1.54148 e+007

TABLE XVII
PERFORMANCE WITH DIFFERENT SEQUENCES—IPD PROBLEM

IPD problem	*Rand 1*	*Rand 2*	*Rand 3*	*Logistic*	*Sinusoid.*	*Gauss*	*Tent*	*Lozi*
Mean Offline	20.2094	20.3764	19.7690	22.3238	19.8055	20.426	20.1855	20.8667
Std.Dev.	2.1473	2.0559	1.9238	2.1730	2.0748	2.3993	2.1778	1.9004
Max.Score	29.1080	29.2560	28.8995	29.5760	29.6340	29.2530	29.0575	29.1115

TABLE XVIII
T-TEST FOR THE $f1$ FUNCTION

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	10.4246	1.665	10.4457	1.678	10.4216	1.677
Rand 2	15.0867	1.663	15.2493	1.678	15.0251	1.677
Rand 3	10.4246	1.665	10.4457	1.678	10.4216	1.677

TABLE XIX
T-TEST FOR THE $f2$ FUNCTION

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	40.0537	1.665	8.5637	1.678	15.1211	1.677
Rand 2	42.713	1.663	11.2587	1.678	17.4643	1.677
Rand 3	40.053	1.665	8.5637	1.678	15.1211	1.677

TABLE XX
T-TEST FOR THE $f3$ FUNCTION

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	60.348	1.665	66.4466	1.678	55.1827	1.677
Rand 2	4.4662	1.663	4.4089	1.678	2.29303	1.677
Rand 3	60.3483	1.665	66.4466	1.678	55.18271	1.677

The t_0 value was computed as and

$$t_0 = \frac{\overline{x} - \overline{y}}{\sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}}} \tag{29}$$

$$\nu = \frac{\left(\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}\right)}{\frac{(S_X^2/n_x)^2}{n_X+1} + \frac{(S_Y^2/n_Y)^2}{n_Y+1}} - 2 \tag{30}$$

TABLE XXI
T-TEST FOR THE $f4$ FUNCTION

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	3,08052	1.665	14,1743	1.678	8,6200	1.677
Rand 2	4,286	1.663	15,1257	1.678	9,5388	1.677
Rand 3	4,1393077	1.665	14,3720	1.678	9,3396	1.677

TABLE XXII
T-TEST FOR THE $f5$ FUNCTION

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	3.7275	1.665	4.69017	1.678	1.94050	1.677
Rand 2	4.3103	1.663	5.8046	1.678	1.94672	1.677
Rand 3	4.62143	1.665	6.4100	1.678	2.0030	1.677

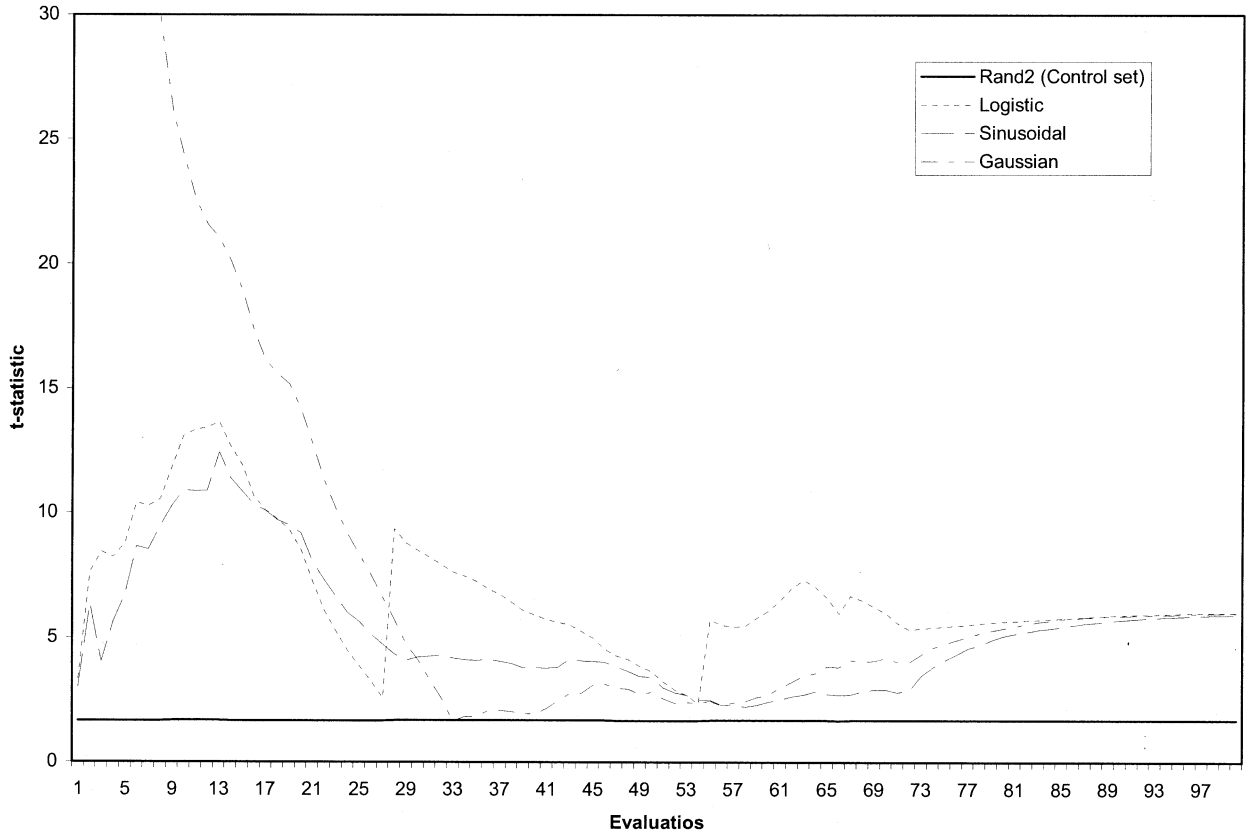

Fig. 7. The t-test on the $f1$ function; control sequence $X(t_{\alpha,\nu})$ is obtained with the best random generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal, and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

the corresponding value of $t_{\alpha,\nu}$ was taken from the table of the t distribution [28] with a level of significance $\alpha = 0.05$.

As shown in Tables XVIII–XXII and Figs. 7–11, the assumption $\mu_X > \mu_Y$ ($t_0 > t_{\alpha,\nu}$) is evidenced for each considered chaotic generator, thus confirming the independence of our results from the choice of the initial population and the fast convergence of the chaotic EAs.

A different set of t-tests were conducted on 50 different TSPs, each with 20 cities with different distances. For these problems, the EAs were run based on *Rand 1*, logistic, sinusoidal, and Gauss map generators.

Each algorithm was run for 100 generations and the t-tests were performed every generation by using the mean fitness values obtained with *Rand 1* for the 50 TSPs as a control sequence X and the corresponding values obtained by using each chaotic generator as sequence Y.

Also, in this case, we tested the hypothesis that $\mu_X > \mu_Y$, corresponding to verify that $t_0 > t_{\alpha,\nu}$. The results obtained are

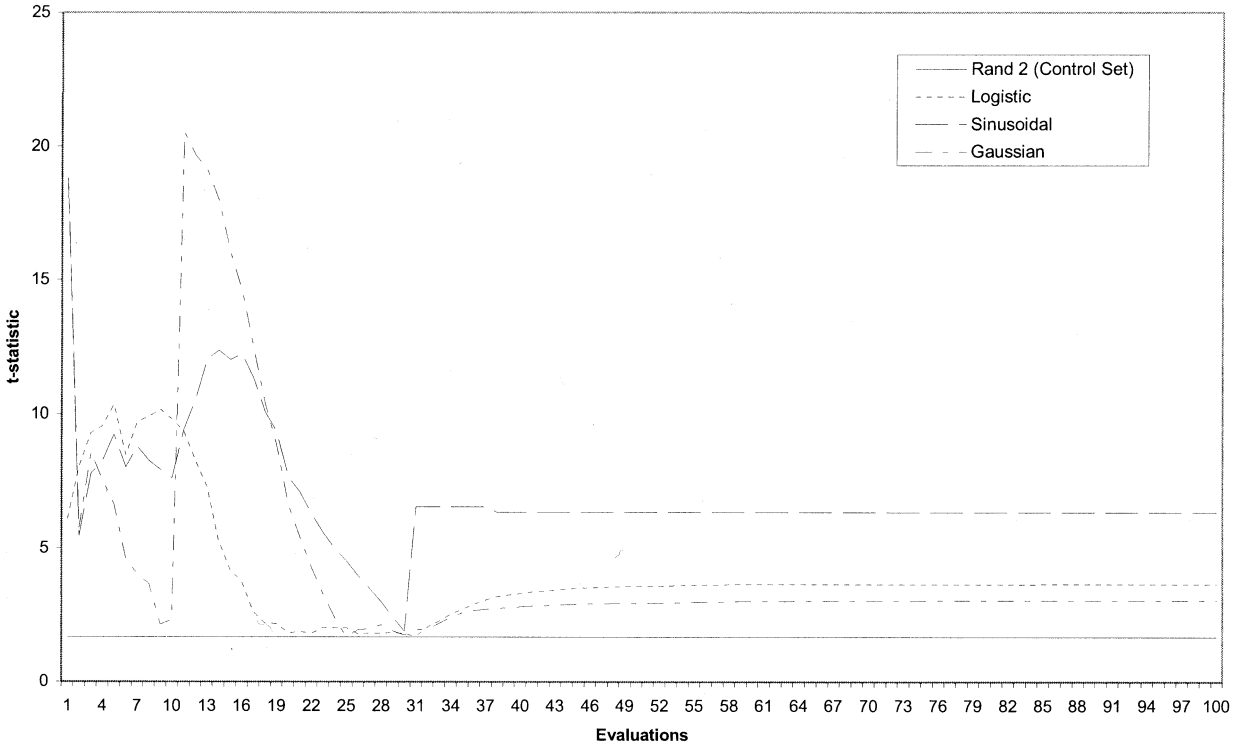

Fig. 8. The t-test on the $f2$ function; control sequence $X(t_{\alpha,\nu})$ is obtained with the best random generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal, and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

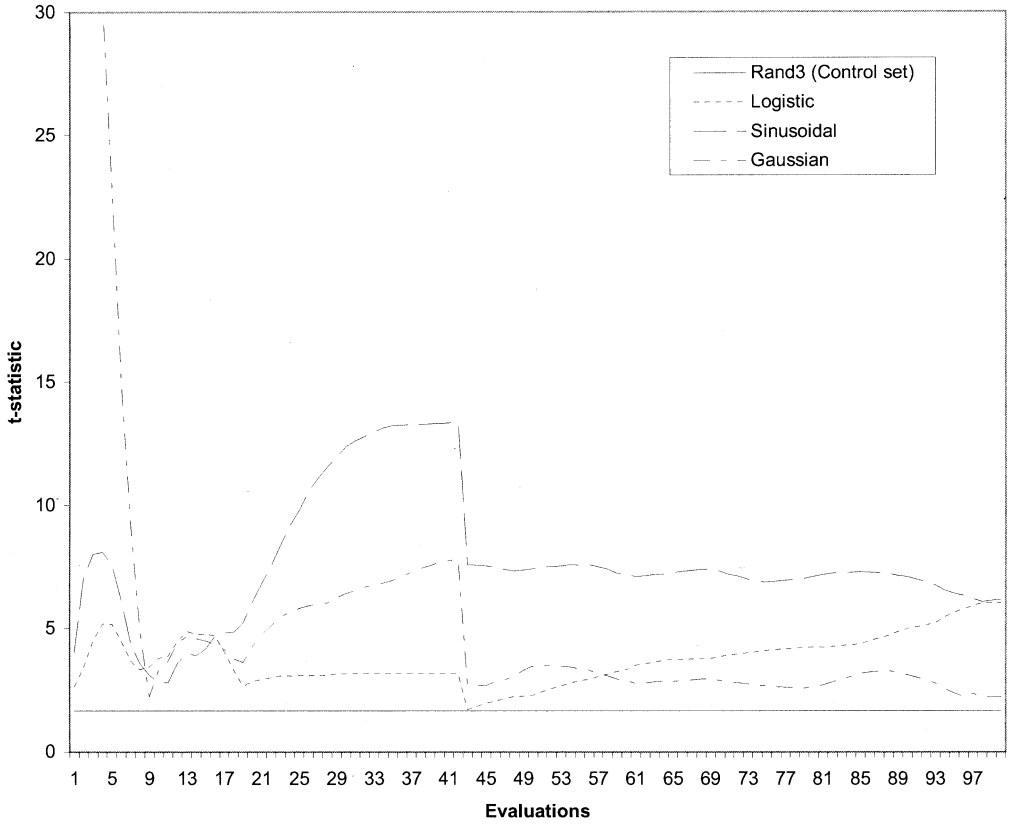

Fig. 9. The t-test on the $f3$ function; control sequence $X(t_{\alpha,\nu})$ is obtained with the best random generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal, and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

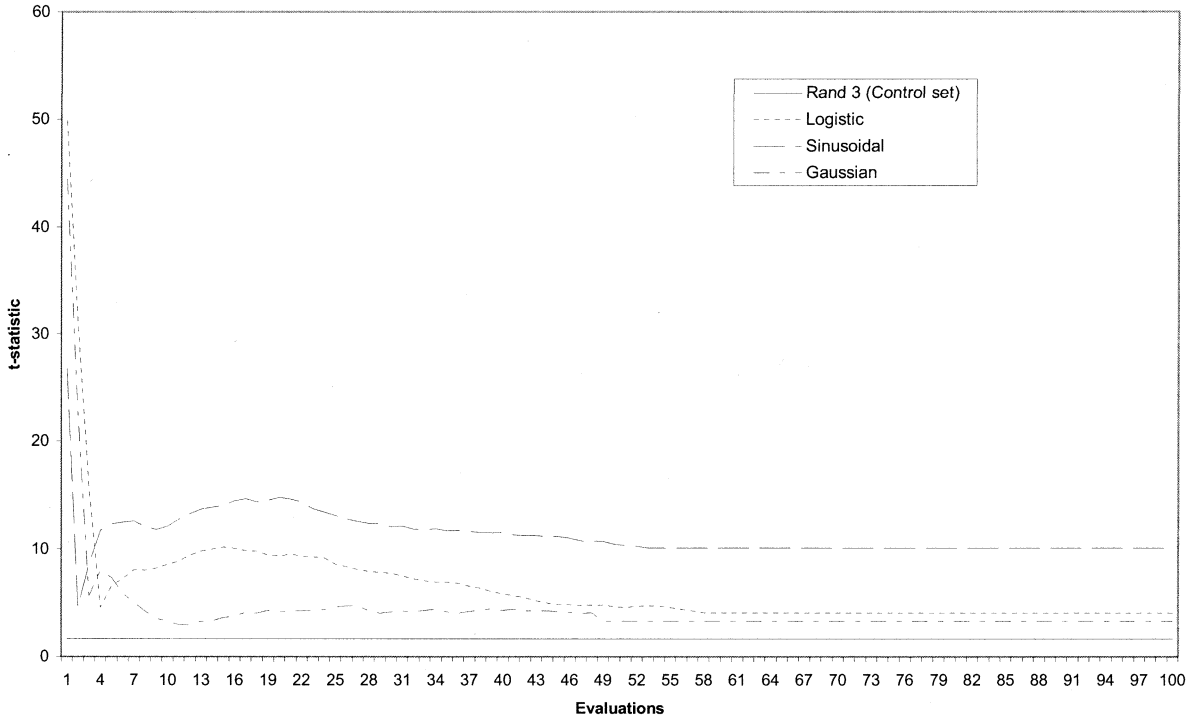

Fig. 10. The t-test on the $f4$ function; control sequence $X(t_{\alpha,\nu})$ is obtained with the best random generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal, and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

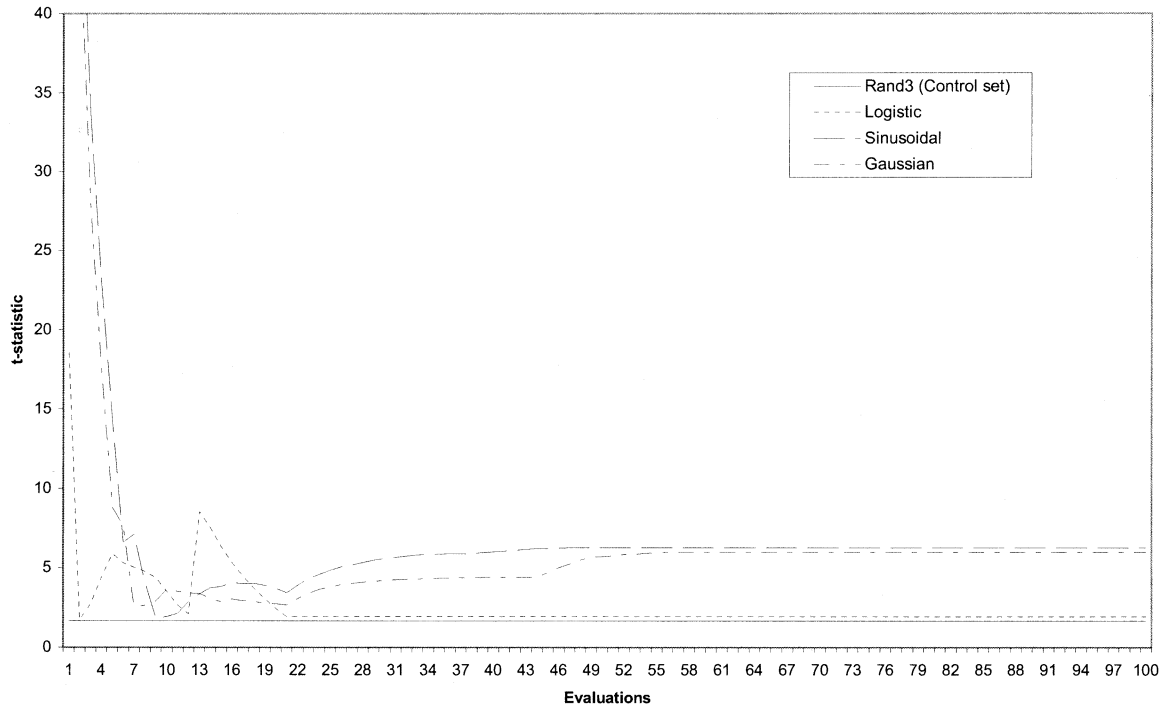

Fig. 11. The t-test on the $f5$ function; control sequence $X(t_{\alpha,\nu})$ is obtained with the best random generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal, and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

shown in Table XXIII and in Fig. 12, where the three trends of the t_0 values and the corresponding three trends of $t_{\alpha,\nu}$, with $\alpha = 0.05$ are depicted versus the number of generations. For all the generations, the hypothesis $t_0 > t_{\alpha,\nu}$ is evidenced, thus confirming that statistically significant fast convergence and better performance was obtained with the chaotic generators.

TABLE XXIII
T-TEST FOR THE TS PROBLEM

Control set	*Gauss*		*Sinusoid.*		*Logistic*	
	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$	t_0	$t_{\alpha,\nu}$
Rand 1	2.6424	1.6725	4.3517	1.6779	4.3517	1.6779
Rand 2	3.1536	1.6725	4.6918	1.6779	4.6918	1.6779
Rand 3	4.1171	1.6725	5.1535	1.6779	5.1535	1.6779

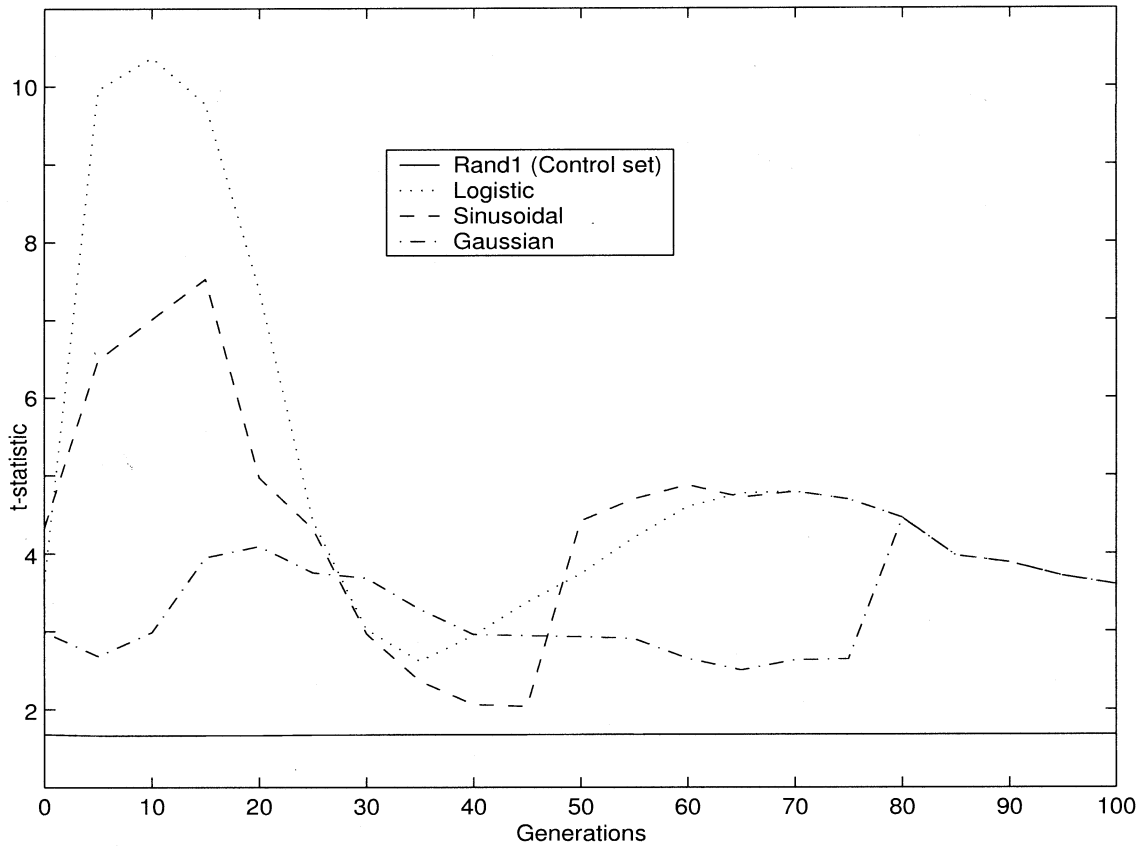

Fig. 12. t-test on the 50 TS problems; control sequence $X(t_{\alpha,\nu})$ is obtained with the best generator. The sequences $Y(t_0)$ are obtained with the generators logistic, sinusoidal and Gauss map. The $t_{\alpha,\nu}$ and t_0 values are reported versus the generations number. The three $t_{\alpha,\nu}$ trends assume very similar values and are indistinguishable in the figure.

VII. CONCLUSION

This paper studied the use of chaotic number generators instead of a random one in EAs. One of the aims of this work is to emphasize how coupling emergent results in different areas, like those of evolutionary computation and complex dynamics, can improve the search for solutions in some optimization problems. In particular, in this paper, numerous examples and statistical tests showed an improvement of the EAs when chaotic sequences were used instead of random processes. The experiments used both a wide class of RNGs and various chaotic time series dynamic generators. The chaotic generators were adopted in the selection procedure, and the crossover and mutation operations, while the initial population was fixed at the same value for each experiment. Four classes of examples were considered to show the suitability of using chaotic number generators instead of standard RNGs. Simulations showed how the best results, in terms of offline performance indexes and minimum number of generations needed, could be reached when these problems were solved by EAs using chaotic generators. Moreover, a statistical analysis using the t-test method was performed in order to validate the improved performance of the EA.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial System*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[2] D. E. Goldberg, *Genetic Algorithm in Search Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[3] Z. Michalewicz, *Genetic Algorithm + Data Structure = Evolution Program*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.
[4] M. D. Vose, *The Simple Genetic Algorithm. Foundation and Theory*. Cambridge, MA: MIT Press, 1999.
[5] W. Banzhaf and C. Reeves, *Foundation of Genetic Algorithm 1–5 Set*. San Mateo, CA: Morgan Kaufmann, Apr. 1999.
[6] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. New York: Wiley, 1998.
[7] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. New York: Wiley, 2000.

[8] A. M. Zalzala and P. Fleming, Eds., "Genetic algorithm in engineering systems," in *IEE Control Engineering*, ser. 55, 1997.
[9] R. Caponetto, M. Criscione, L. Fortuna, D. Occhipinti, and L. Occhipinti, "Synthesis of a programmable chaos generator, based on CNN architectures, with applications in chaotic communication," in *Proc. CNNA '98*, London, U.K., Apr. 14–17, 1998, pp. 124–129.
[10] M. Bucolo, R. Caponetto, L. Fortuna, and M. G. Xibilia, "How the chua circuit allows to model population dynamics," presented at the Proc. NOLTA '98, La Regent, Crans-Montana, Switzerland, Sept. 14–17, 1998.
[11] H. Nozawa, "A neural network model as globally coupled map and application based on chaos," *Chaos*, vol. 2, pp. 377–386, 1992.
[12] L. Chen and K. Aihara, "Global searching ability of chaotic neural networks," *IEEE Trans. Circuit Syst. I*, vol. 46, no. Aug., pp. 974–993, 1999.
[13] L. Wang and K. Smith, "On chaotic simulated annealing," *IEEE Trans. Neural Networks*, vol. 9, pp. 716–718, July 1998.
[14] P. Arena, R. Caponetto, L. Fortuna, A. Rizzo, and M. La Rosa, "Self organization in non recurrent complex system," *Int. J. Bifurcation and Chaos*, vol. 10, no. 5, pp. 1115–1125, 2000.
[15] J. Determan and J. A. Foster, "Using chaos in genetic algorithm," in *Proceedings of the 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1999, vol. 3, pp. 2094–2101.
[16] G. Manganaro and J. Pineda de Gyvez, "DNA computing based on chaos," in *Proc. 1997 IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1997, pp. 255–260.
[17] L. Juan, C. Zixing, and L. Jianqin, "Premature convergence in genetic algorithm: Analysis and prevention based on chaos operator," in *Proc. 3rd World Congress Intelligent Control Automation*, Hefei, China, June 28–July 2, 2000, pp. 495–499.
[18] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishanm, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA: SIAM Books, 1994.
[19] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[20] S. Park and K. Miller, "Random number generators: Good ones are hard to find," *Commun. ACM*, vol. 31, pp. 1192–1201, 1988.
[21] D. Knuth, "Seminumerical algorithms," in *The Art of Computer Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1981, vol. 2.
[22] P. L'Ecuyer, "Efficient and portable combined random number generators," *Commun. ACM*, vol. 31, pp. 742–774, 1988.
[23] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic System*. Berlin, Germany: Springer-Verlag, 1989.
[24] H. Peitgen, H. Jurgens, and D. Saupe, *Chaos and Fractals*. Berlin, Germany: Springer-Verlag, 1992.
[25] R. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulating Annealing*, L. D. Davis, Ed. San Mateo, CA: Morgan Kaufmann, 1987, pp. 32–41.
[26] D. Fogel, *Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1995.
[27] Galib. GaLib 2.4.2, a C++ library of genetic algorithm components developed by Matthew Wall. [Online]. Available: http://lancet.mit.edu.
[28] D. C. Montgomery, *Statistical Quality Control*. New York: Wiley, 1996.

Riccardo Caponetto (S'93–M'96) was born in Catania, Italy, in 1966. He received the electronic degree in 1991 and the Ph.D. degree from the University of Catania, Catania, Italy, in 1994.

In 1994, he was working as a Researcher at STMicroelectronics, Catania, Italy. From 1995 to 1998, he was a Professor of industrial robotics at the University of Messina, Messina, Italy. Since 2001, he has been an Assistant Professor at the D.I.E.E.S of the Engineering Faculty, University of Catania. His interest include soft computing techniques, modeling, and control of complex systems and robotics.

Luigi Fortuna (M'90–SM'99–F'00) is a Full Professor of system theory at the University of Catania, Catania, Italy, since 1994. He has published more than 250 technical papers and is coauthor of six books including *Cellular Neural Networks* (New York: Springer-Verlag, 1999). He holds several U.S. patents. His scientific interests include nonlinear science and complexity, chaos, cellular neural networks with applications in bioengineering.

Dr. Fortuna is Chair of IEEE Technical Committees in Cellular Neural Networks and Array Computing.

Stefano Fazzino was born in Siracusa, Italy, in May 1974. He received the degree of Information Technology Engineering (*cum laude*) from the University of Catani, Catani, Italy, in 1998. In 2000, he joined the DEES Department, University of Catania, working toward the Ph.D. degree in electronic and automatic engineering.

He worked as a Software Design Engineer, Nokia Networks, Catania, Italy. His current research topics are evolutionary algorithms and nonlinear systems.

Maria Gabriella Xibilia (S'93–M'01) was born in Catania, Italy, in 1966. She received the electronic degree in 1991 and the Ph.D. degree in 1994 from the University of Catania, Catania, Italy.

Since 1998, she has been an Assistant Professor in the Mathematics Department, University of Messina, Messina, Italy. Her interest include soft computing, virtual sensors, automatic control, nonlinear systems identification and control.