An Enhanced Hybrid Chaotic Algorithm using Cyclic Coordinate Search and Gradient Techniques

Un algoritmo caótico híbrido mejorado de búsqueda por coordenadas cíclicas y técnicas de gradiente

Juan David Velásquez[a]

PALABRAS CLAVES

Algoritmos de optimización de caos, funciones no lineales, minimización, métodos cíclicos de búsqueda por coordenadas.

RESUMEN

En este artículo se presenta un algoritmo híbrido caótico que usa una búsqueda cíclica mejorada a lo largo de cada eje y el algoritmo BFGS para optimizar funciones no lineales. El método propuesto es una poderosa técnica de optimización; esto es demostrado al optimizar cuatro funciones benchmark con 30 dimensiones. La metodología propuesta es capaz de converger a una mejor solución, y más rápido que el algoritmo tradicional de optimización basado en caos, y otras técnicas competitivas.

KEY WORDS

Chaos optimization algorithms, nonlinear test functions, minimization, cyclical coordinates search methods.

ABSTRACT

In this paper, we present a hybrid chaotic algorithm using a chaotic enhanced cyclical search along each axis and the BFGS method for optimizing nonlinear functions. The proposed method is a powerful optimization technique; this is demonstrated when four nonlinear benchmark functions with 30 dimensions are minimized using the proposed technique. Using this methodology we are able to find a better and faster solution than the traditional chaos optimization algorithm and other competitive techniques.

a PhD en Ingeniería. Profesor asociado. Universidad Nacional de Colombia. Medellín, Colombia. ✉ jdvelasq@unal.edu.co

INTRODUCTION

The use of heuristic algorithms for optimizing nonlinear functions is an important and growing field of research [1]; this is due to the fact that in mathematics, engineering and economics, the maximization or minimization of highly nonlinear functions is a common, important and challenging problem. This difficulty is explained by the complexity of the objective function, the restrictions imposed on the problem, the presence of the so-called multiple local minima and the limitations of many optimization methodologies [1]. It is a well-known fact that gradient-based optimization algorithms are trapped within local optimum points.

Coordinate-based search algorithms are some of the more traditional classical methods [2] [3] [4] for function minimization. They seek the local optimum sequentially along a single axis at a time while the values for other axes are fixed.

Recently, chaos theory has been used in the development of novel techniques for global optimization [5] [6] [7] [8], and particularly, in the specification of chaos optimization algorithms (COA) based on the use of numerical sequences generated by means of a chaotic map [5] [7] [9] [10] instead of random number generators. However, recent trends are about the hybridization of the COA with other well established techniques as gradient-based methods [11], genetic algorithms [12], particle swarm optimization [13] [14] [15] [16] [17] [18], differential evolution [19], clonal algorithms [20], artificial immune systems [21] [22], bee colony algorithms [23] and simulated annealing [24].

The nonlinear optimization problems is stated as min $f(\mathrm{x})$ subject to $\mathrm{L} \leq \mathrm{x} \leq \mathrm{U}$, where x, L, U are vectors of $n \times 1$, and $f(\,)$ is a nonlinear function such that $f{:}R^n \rightarrow R$. This problem is the same as those formulated in other optimization algorithms for which it is necessary: (a) to restrict the search space due to limitations of the algorithm, as the Monte Carlo optimization [25]; or (b) to transform the representation of the solution into real values, as tabu search [26] [27] and genetic algorithms [28]. However, the problem definition used here is not restrictive in relation to the application cases, and our algorithm would be applied to solve problems with complex restrictions using penalizing functions, among others methods. For example, as proposed in [29], we can minimize problems with restrictions using a new function $F(\mathrm{x})$ defined as:

$$\mathrm{F(x)} = \begin{cases} f(\mathrm{x}) & \mathrm{x} \in \Omega \\ M_e + d_e & \mathrm{x} \notin \Omega \end{cases} \qquad (1)$$

where Ω represents the feasible region, and M_e is the maximum value inside of Ω. The penalization term is defined as $d_e = \max\{H_1 \times |R_1|, H_2 \times |R_2|, \ldots\}$; where H_i is one if the i-th restriction is violated, and zero otherwise; R_i is the magnitude of the violation of the i-th restriction. Obviously, in the restrictions R_i we do not consider the limits imposed by $\mathrm{L} \leq \times \leq \mathrm{U}$.

The aim of this paper is to present a novel chaos optimization algorithm based on cyclical coordinate search. The paper is organized as follows: the next section summarizes the traditional chaos optimization algorithm; following that, we present a new methodology based on chaos theory; next, we analyze the behavior of the proposed algorithm when four well-known nonlinear test functions are optimized. Finally, we provide some conclusions.

TRADITIONAL CHAOS OPTIMIZING ALGORITHM (COA)

Chaos is understood as the complex, bounded and unstable behavior caused by a simple deterministic nonlinear system or chaotic map, such that, the generated sequences are quasi-random, irregular, ergodic, semi-stochastic and very sensible to the initial value [30]. The use of chaotic sequences instead of quasi-random number generators seems to be a powerful strategy for improving many traditional heuristic algorithms, and their main use is in escape of local minima points [31]. The logistic map is a very common one-dimensional non-invertible model for generating chaotic sequences (for n=1,2, ...; and $0 < \lambda \leq 4$):

$$Y_{n+1} = \lambda Y_n (1-Y_n) \qquad (2)$$

Where $Y_n \in [0,1]$ and $Y_n \notin \{0., 0.25, 0.50, 0,75, 1.0\}$.

The most elementary optimization procedure [5] consists in generating candidate points x_c inside of the feasible region $L \leq x \leq U$; the optimum, x_1, is the candidate point with the lowest value of $f(x_c)$. The process is schematized in Figure 1, from line 02 to line 08. Candidate points x_c (line 03) are generated in the domain [L, U] by means of the vector of chaotic sequences Y_1. In order to do this, the *i*-th component of Y_1, is mapped linearly to the interval [L(*i*), U(*i*)]. In the algorithm presented in Figure 1, we assume that the components of Y_1 are restricted to the interval [0, 1] as occurs for the logistic map. At each iteration, a new vector of chaotic sequences is generated using the chaotic map $H(\,)$(line 7); in our case, $H(\,)$ is the logistic map defined in (1) but another maps would also be used. The current local optimum x_1 is updated (line 6) in each iteration. The algorithm made up by lines from 01 to 08 is the so-called first carrier wave and it is similar to the Monte Carlo optimization technique, which converges slowly and obtains the global optimum with low probability.

The first carrier wave is used to obtain a good point for the refining phase or second carrier wave [5] described by the lines 09 to 16. Y_2 is other vector of chaotic sequences, and *r* is a scalar parameter related to the radius of search around of x_1. The value of *r* is decreased by means of a function $P(\,)$ (line 14) in each iteration. Each candidate point is generated inside the hypercube $[x_1-r, x_1+r]$, since each component of Y_2 (with domain [0, 1]) is mapped to the interval [-*r*, *r*]. The local optima is updated every time that a better point is found (line 13), such that, the procedure continues seeking in the neighborhood of the new optimum point. The search procedure is similar in some fashion to the simulated annealing technique when ascending movements are not allowed. An initial value for $r = 0.1$ (line 9) is proposed in [32], and $P(r)=\lambda r$, with $0<\lambda<1$, as a useful scheme for reducing *r* (line 14). In addition, it is necessary to define the minimal value for *r*, always as always $r>0$.

PROPOSED METHODOLOY

In this section we describe the methodology presented in Figure 2. We begin with an initial point drawn from a compact domain (line 02); in line 02, *u* is a uniform random number in the interval. The basis of our method is the search along each coordinate axis (line 05). Using the current best solution, we obtain a new candidate point changing the *i*-th component for a random value inside of the interval centered in the current best value (for the *i*-th component) with radius *r(i)* (line 09); the search radius *r* is reduced (line 17) by subtracting a fixed value, α, after each complete cycle. We repeat this sampling process K_2 times (line 06), and each time we obtain a better point, the best current solution is updated (from lines 11 to 13). The complete cycle is repeated K_1 times (line 03). Once we complete a cycle over the *n* components of x, we refine the current best solution using the BFGS gradient-based optimization algorithm (line 18). The function *g( )* in line 18 calls for the implementation of the BFGS method.

We assume that the chaotic map $H(\,)$ is bounded in the interval [0, 1] (line 07); for this reason, we use the transformation $[2Y(i)-1]$ to convert the interval [0, 1] to [-1, 1]. The draw mechanism specified in line 09 is inspired by the simulated annealing technique; however, sampling along each axis seems to be more effective than the traditional simulated annealing technique.

In our experiments, we found that the proposed algorithm works fine for *r* initialized in a value of 0.1. The parameter α is a function of the final search ratio and this is calculated, such that, $r = 0.0001$ in the last iteration of the main cycle (line 03). In other words, the search ratio *r* is reduced from an initial value specified in line 01 to the last value, reached in the last cycle, subtracting in each iteration (line 03) a small fixed value α.

The current implementation of the methodology was made in the R language for statistical computing. We use the implementation of the BFGS method available in the primitive function optim. We use the default values for the main parameters of the optim function.

```
01   initialize Y1   # first carrier wave #
02   for (m1=1, ...; M1) {
03        let xc=L+Y1(U-L)
04        if (m1==1) let xl=xc
05        let Δf=f(xc)-f(xl)
06        if (Δf<0) let xl=xc
07        let Y1=H(Y1)
08   }
09   initialize r and Y2   # second carrier wave #
10   for (m2=1, ..., M2) {
11        let xc=xl+r (2Y2-1)
12        let Δf=f(xc) - f(xl)
13        if (Δf<0) {let xl=xc}
14        let r =P(r)
15        let Y2=H(Y2)
16   }   # end of algorithm
```

Figure 1. Chaos optimization algorithm

```
01        initialize Y, K1, K2, α, r
02        let xl=L+u×(U-L)
03        for (k1=1, ..., K1) {
04             let found = FALSE
05             for (i = 1, …, n) {
06                  for (k2=1, ..., K2) {
07                       let Y(i)=H(Y(i))
08                       let xc =xl
09                       let xc(i)=xc(i)+r[2Y(i)-1]
10                       let Δf=f(xc)-f(xl)
11                       if (Δf<0) {
12                            let xl=xc
13                            let found=TRUE
14                       }
15                  }
16             }
17             let r =r -α
18             if (found==TRUE), xl=g(xl)
19        } # end of algorithm
```

Figure 2. Proposed enhanced chaos optimization algorithm

SIMULATIONS

A major problem in the validation of chaos optimization algorithms is the use of classical test functions for a low number of dimensions, so that, it is not possible to evaluate the real power of these algorithms. For example, in ref. [32] only functions with 2 or 3 dimensions are tested, while in ref. [33], the Camel and Shaffer two-dimensional functions are evaluated. In our study, we overcome this limitation using a major number of dimensions (30) and comparing with other heuristic optimization algorithms.

The algorithms described in previous section are applied to the following test functions in order to better understand its behavior and to clarify its efficiency:

Sphere: $$f(\mathrm{x}) = \sum_{i=1}^{N} x_i^2$$

DeJongF4: $$f(\mathrm{x}) = \sum_{i=1}^{N} i \cdot x_i^2$$

Griewank: $$f(\mathrm{x}) = 1+\sum_{i=1}^{N} \frac{x_i^2}{4000} + \prod_{i=1}^{N} \cos \frac{x_i}{\sqrt{i}}$$

Rastrigin: $$f(\mathrm{x}) = \sum_{i=1}^{N} [10+x_i^2 - 10 \cos 2\pi x_i]$$

The first two functions (Sphere and DeJongF4) have a unique global minimum. Griewank function has many irregularities but it has only one unique global minimum. The Rastrigin function has many local optimal points and one unique global minimum. For this study, N was fixed in 30 (dimensions). Table 1 resumes the global optimum, the function value at global optimum and the search range used for each test function. Figure 3 presents the plot for each test function. For each function and each algorithm considered, we use 50 random start points (50 runs); each run was limited to 15000 evaluations of the test function.

Name	Global optimum	Function value at optimum	Search range
Sphere	(0,0, ..., 0)	0.0	[-50, 50]
DeJongF4	(0,0, ..., 0)	0.0	[-20, 20]
Griewank	(0,0, ..., 0)	0.0	[-600, 600]
Rastrigin	(0,0, ..., 0)	0.0	[-5.12, 5.12]

Table 1. Test functions used in this study

First, we use, as a benchmark, the traditional chaos optimization algorithm (COA), which is made up by two cycles as described in Figure 1. We use 5000 iterations for the first carrier wave, in order to generate a good starting point for the second part. Ten thousand iterations were used for the second wave carrier. Figure 4 presents the best run for each function. It is noted that the first part is not able to reach a good initial point; and there is a vertical line indicating the starting point of the second wave carrier. In Table 2, the results for each function and each algorithm are presented.

Second, we apply our methodology to the test functions. For this, we use 5 main cycles (K_1 parameter in line 03), and 100 tries for each axis (K_2 parameter in line 06). The initial value of r is 0.1(L-U) and the final value is 0.0001(L-U). Values for L and U are presented as the search range column in Table 1. The obtained results are presented in Table 2. In all cases, our algorithm is an improvement on the COA approach, in terms of the best solution found, the best mean value and the standard deviation of the best solutions. Our algorithm is especially successful for the Rastrigin function, where the COA gave a very poor solution.

Figure 5 shows the best run for our algorithm. In comparison with Figure 4, our algorithm find lower values of the objective function faster than COA. Moreover, for the Rastringin function, our algorithm is able to quickly escape from the local optimal points unlike the COA algorithm, which has the same value of the objective function for many iterations.

We also compared our methodology against other approaches. In Table 2, we report the results presented in [34] for the classical evolutionary programming methodology (CEP) and the fast evolutionary programming (FEP). In [34], CEP and FEP algorithms are used for optimizing the sphere, Rastrigin and Griewank functions. These algorithms use a popula-

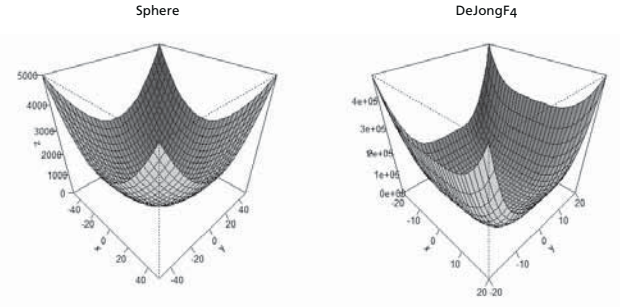

Figure 3. Plots of test functions used in this study

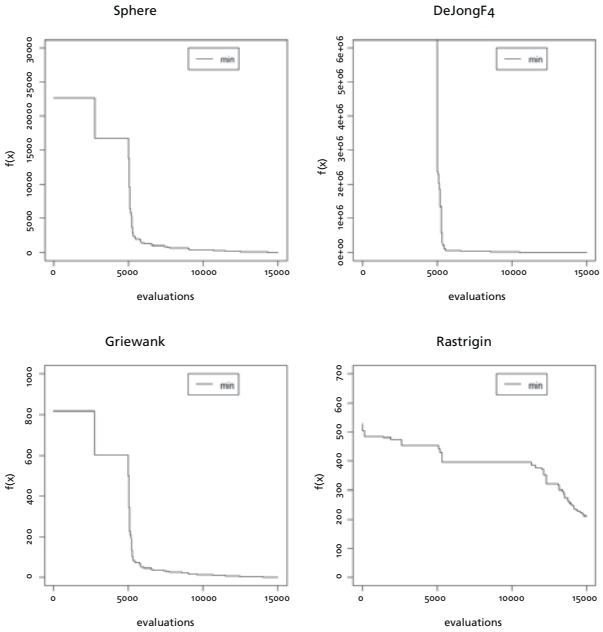

Figure 4. Best run for basic chaos optimization algorithm

Algorithm	Best value	Mean best value	Std. Dev
Sphere function			
COA	0.3777	1.1870	0.4956
This study	9.0481×10^{-38}	**1.3733×10^{-37}**	**2.7020×10^{-38}**
CEP	-	2.2×10^{-4}	5.9×10^{-4}
FEP	-	5.7×10^{-4}	1.3×10^{-4}
GEBOUD	-	1.453×10^{-32}	9.997×10^{-32}
DeJongF4 function			
COA	0.0214	0.2270	0.2413
This study	9.4160×10^{-15}	**9.6814×10^{-14}**	**7.6192×10^{-14}**
Griewank function			
COA	0.9812	1.0392	0.0201
This study	0.0025	0.0128	0.0133
CEP	-	8.6×10^{-2}	1.2×10^{-1}
FEP	-	1.6×10^{-2}	2.2×10^{-2}
rgenoud [35]	-	**7.994×10^{-17}**	**1.209×10^{-16}**
Rastrigin function			
COA	68.0986	127.2812	33.6033
This study	0.0000	**0.0199**	**0.1407**
CEP	-	89.0	23.1
FEP	-	4.6×10^{-2}	1.2×10^{-2}
rgenoud [35]	-	2.786	1.864

Table 2. Comparison of algorithms. All results have been averaged over 50 runs. "Best value" indicates the minimum value of the objective function over 50 runs. "Mean best value" indicates the mean of the minimum values. "Std. Dev" is the standard deviation of the minimum values.

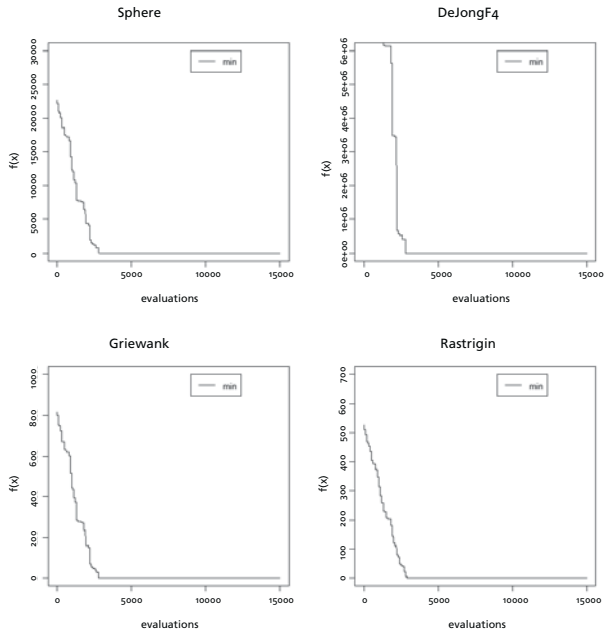

Figure 5. Best run for chaos-based coordinate search algorithm

tion of 100 individuals and 1.500, 2.000 and 5.000 generations; that gives, a total of 15.000, 20.000 and 50.000 calls to the objective function. In comparison, we use only 15.000 calls to the objective function for all functions. The results reported in [34] are reproduced in Table 2. For the considered functions, the proposed algorithm has a lower mean best value than the evolutionary programming technique. Also, we compare our methodology against the rgenoud package [35] that combines evolutionary search algorithms with gradient based Newton and quasi-Newton methods for optimizing highly nonlinear problems. In Table 2, we report the results published in [35] for 30 generations and a population of 5000 individuals (15.000 calls to the objective function); in this case, rgenoud is better than our methodology for the Griewank function. Thus, we conclude that our approach is competitive with other well established optimization techniques.

CONCLUSIONS AND FUTURE WORK

In this paper, we present a new optimization methodology inspired by coordinate search methods, chaos optimization algorithms and gradient-based techniques. For testing our approach, we use 4 well known nonlinear benchmark functions. The presented evidence allows us to conclude that the proposed methodology is fast and converges to good optimal points, at least for the proposed functions.

Our main conceptual contributions are: (a) to use a sampling mechanism in the coordinate search methods based in chaos theory; (b) to change the search strategy in the chaos optimization algorithm, incorporating a coordinate search strategy; and (c) to refine the final solution using the BFGS gradient-based method.

However, further research is needed to gain more confidence and a better understanding of the proposed methodology. It is necessary: (a) To evaluate the proposed algorithm for a major number of test functions; (b) To analyze the behavior of our methodology when it is applied to real world problems, like the training of neural networks; (c) To prove the algorithm with other types of chaotic maps and techniques for sampling.

BIBLIOGRAPHY

[1] P. M. Pardalos & M. G. C. Resende, (Ed).
Handbook of Applied Optimization. New York: Oxford University Press, 2002.

[2] M. S. Bazaraa, H. D. Sherali & C. M. Shetty.
Nonlinear Programming: Theory and Algorithms. 2nd Edition. N.J.: John Wiley and Sons Inc, 1996.

[3] E. Fermi & N Metropolis.
Numerical Solutions of a minimum problem. Technical Report Los Álamos, LA-1492, 1952.

[4] R. Hooke & T.A. Jeeves.
"«Direct search» solution of numerical and statistical problems". *J. Assoc. Computer.* Vol. 8, 1961, pp. 212-229.

[5] B. Li & W.S. Jiang.
"Chaos optimization method and its application". *Journal of Control Theory and Application*, Vol. 14, No. 4, 1997, pp. 613-615.

[6] B. Li & W.S. Jiang.
"Optimizing complex function by chaos search". *Cybernetics and Systems*, Vol. 29, No. 4, 1998, pp. 409–419.

[7] C. Choi & J.J. Lee.
"Chaotic local search algorithm". *Artificial Life and Robotics*, Vol. 2, No. 1, 1998, pp. 41-47.

[8] C. Zhang, L. Xu, & H. Shao.
"Improved chaos optimization algorithm and its application in nonlinear constraint optimization problems". *Shanghai Jiaotong Daxue Xuebao, Journal of Shanghai Jiaotong University*, Vol. 34, No. 5, pp. 2000593-595, 599.

[9] M. S. Tavazoei & M. Haeri.
"An optimization algorithm based on chaotic behavior and fractal nature". *Journal of Computational and Applied Mathematics*, Vol. 206, No. 2, 2007, pp.1070-1081.

[10] D.Yang, G. Li & G. Cheng.
"On the efficiency of chaos optimization algorithms for global optimization". *Chaos, Solitons and Fractals*, Vol. 34, 2007, pp. 1366–1375.

[11] Y.X.,Ou-Yang, M. Tang, S.L. Liu, J.X. Dong.
"Combined BFGS-chaos method for solving geometric constraint". *Zhejiang Daxue Xuebao (Gongxue Ban)/Journal of Zhejiang University (Engineering Science)*, Vol. 39, No. 9, 2005, pp. 1334-1338.

[12] C.G. Fei & Z.Z. Han.
"A novel chaotic optimization algorithm and its applications". *Journal of Harbin Institute of Technology (New Series)*, Vol. 17, No. 2, 2010, pp. 254-258.

[13] B. Liu, L. Wang, Y.H. Jin, F. Tang & D.X. Huang.
"Improved particle swarm optimization combined with chaos". *Chaos, Solitons and Fractals*. Vol. 25, No. 5, September 2005, pp. 1261-1271.

[14] R.Q., Chen & J.S. Yu.
"Study and application of chaos-particle swarm optimization-based hybrid optimization algorithm". *Xitong Fangzhen Xuebao / Journal of System Simulation*, Vol. 20, No. 3, 2008, pp. 685-688.

[15] H.A. Hefny & S.S. Azab (2010).
"Chaotic particle swarm optimization". *2010 7th International Conference on Informatics and Systems, INFOS2010*. Cairo, 28 March 2010 through 30 March 2010. Category number CFP1006J-ART. Code 80496.

[16] Z. Chen & X.
Tian. "Artificial fish-swarm algorithm with chaos and its application". *2nd International Workshop on Education Technology and Computer Science, ETCS 2010*. Vol. 1, 2010, Article number 5458996, pp.ages 226-229.

[17] H.J Meng, P. Zheng, R.Y. Wu, X.-J. Hao, Z. Xie.
"A hybrid particle swarm algorithm with embedded chaotic search". *2004 IEEE Conference on Cybernetics and Intelligent Systems*, 2004, pp. 367-371.

[18] B. Alatas, E. Akin & A. B. Ozer.
"Chaos embedded particle swarm optimization algorithms". *Chaos, Solitons and Fractals*, Vol. 40, No. 4, 2009, pp. 1715-1734.

[19] Z.Y. Guo, L.Y. Kang, B. Cheng, M. Ye, B.G. Cao.
"Chaos differential evolution algorithm with dynamically changing weighting factor and crossover factor". *Harbin Gongcheng Daxue Xuebao/Journal of Harbin Engineering University*, Vol. 27 (SUPPL.), 2006 pp. 523-526.

[20] H. Du, M. Gong, R. Liu, L. Jiao.
"Adaptive chaos clonal evolutionary programming algorithm". *Science in China, Series F: Information Sciences*, Vol. 48, No. 5, 2005, pp. 579-595.

[21] X.Q. Zuo & Y.S. Fan.

"Chaotic-search-based immune algorithm for function optimization". *Kongzhi Lilun Yu Yinyong/Control Theory and Applications*, Vol. 23, No. 6, 2006, pp. 957-960+966.

[22] X.Q. Zuo & S.Y. Li.

"The chaos artificial immune algorithm and its application to RBF neuro-fuzzy controller design". *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, 2003, pp. 2809-2814.

[23] B. Alatas.

"Chaotic bee colony algorithms for global numerical optimization". *Expert Systems with Applications*, Vol. 37, No. 8, 2010, pp. 5682-5687.

[24] J. Mingjun & T. Huanwen.

"Application of chaos in simulated annealing". *Chaos, Solitons and Fractal*s. Vol. 21, No. 4, August 2004, pp. 933-941.

[25] H. Niederreiter & K. McCurley.

"Optimization of functions by quasi-random search methods". *Computing*. Vol. 22, No. 2, 1979, pp. 119-123.

[26] F. Glover.

"Tabu Search — Part I", *ORSA Journal on Computing*. Vol. 1, No. 3, 1989, pp. 190-206.

[27] F. Glover.

"Tabu Search — Part II", *ORSA Journal on Computing*. Vol. 2, No. 1, 1990, pp. 4-32.

[28] D. E. Goldberg.

Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA.: Kluwer Academic Publishers, 1989.

[29] F. Hoffmeister and J. Sprave

"Problem-independent handling of constraints by use of metric penalty functions". *Proceedings of the 5th Conference on Evolutionary Programming (EP '96)*, L. J. Fogel, P. J. Angeline, and T. Bäck (Eds.). Cambridge, UK: MIT Press, February 1996, pp. 289–294.

[30] S.H. Strogatz.

Nonlinear Dynamics and Chaos. Massachussetts: Perseus Publishing, 2000.

[31] R. Caponetto, L. Fortuna, S. Fazzino & M.G. Xibilia.

"Chaotic sequences to improve the performance of evolutionary algorithms". *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 3, 2003, pp. 289–304.

[32] H. Lu, H. Zhang & L. Ma.

"A new optimization algorithm based on chaos". *Journal of Zhejiang University [Science A]*, Vol. 7, No. 4, 2006, pp. 539–542.

[33] T. Quang, D. Khoa & M. Nakagawa.

"Neural Network based on Chaos". *International Journal of Computer, Information and Systems Science, and Engineering*, Vol. 1, No. 2, 2007, pp. 97-102.

Recibido 18 de noviembre de 2009, modificado 21 de noviembre de 2010, aprobado 11 de diciembre de 2010.