ON COMPARING DIFFERENT CHAOTIC MAPS IN DIFFERENTIAL EVOLUTIONARY OPTIMIZATION

Mohamed F. El-Santawy, A. N. Ahmed

Institute of Statistical Studies and Research, Cairo University, Egypt

ABSTRACT: This paper presents a comparison between new approaches introducing different chaotic maps with ergodicity, irregularity, and the stochastic property in Differential Evolution algorithm (DE). The members of the new family so-called Chaotic Differential Evolution (CDE) algorithms employ chaos in order to improve the global convergence by escaping the local solutions.
KEYWORDS: Chaos, Chaotic Maps, Differential Evolution Algorithm, Optimization.

1. INTRODUCTION

In order to overcome the shortcomings of traditional mathematical techniques, nature-inspired soft computing algorithms have been introduced. Several evolutionary or meta-heuristic algorithms have since been developed which combine rules and randomness mimicking natural phenomena. Many researchers have recently studied these meta-heuristic algorithms to solve various optimization problems.
Chaos is a kind of characteristic of nonlinear systems and it has been extensively studied and applied in many fields. Although it appears to be stochastic, it occurs in a deterministic nonlinear system under deterministic conditions. Chaos has been extended to various optimization areas like in [EA11], [EA12] because it can more easily escape from local minima than other stochastic optimization algorithms. Recently, chaotic sequences have been adopted instead of random sequences and very interesting and somewhat good results have been shown in many applications.
Differential evolution (DE) is a population based stochastic search algorithm [SP95], and has been successfully applied to solve complex problems including linear and nonlinear, Uni-modal and Multi-modal functions. Over the recent years, DE has been successfully applied to different subjects such as reservoir system optimization [RK07], optimal design of shell and tube heat exchangers [BM07].
The main goal of this paper is to compare different Chaotic Differential Evolution (CDE) algorithms using different chaotic maps to asses the impact of incorporating them in the classical algorithm. This paper is structured as following: Section 2 is made for Chaos, section 3 is made for Chaotic maps, section 4 is devoted to Differential Evolution approach, the proposed algorithms are illustrated in section 5, experiments and simulation results are shown in section 6, and finally the conclusion is presented.

2. CHAOS

In recent years, the theories and applications of nonlinear dynamics, especially of chaos, have drawn more and more attention in many fields. One is chaos controlling, and synchronization. One is chaos controlling, and synchronization. Another field is the potential applications of chaos in various disciplines including optimization.
Mathematically, chaos is randomness of a simple deterministic dynamical system and chaotic system may be considered as sources of randomness [Ala10].
A chaotic map is a discrete-time dynamical system

$$x_{n+1} = f(x_n), \quad 0 < x_n < 1, \ \ n = 0,1,2,\ldots \qquad (1)$$

running in the chaotic state. The chaotic sequence $\{x_n : n = 0,1,2,\ldots\}$ can be used as spread-spectrum sequence and as a random number sequence. One-dimensional noninvertible maps are the simplest systems with capability of generating chaotic motion [Ott02].

3. CHAOTIC MAPS

In this section, some well-known maps are introduced. Later on, these maps are used in the Chaotic Differential Evolution (CDE) approaches that will be compared.

3.1. Logistic map

In 1976, Robert May pointed out that the logistic map led to chaotic dynamics [May76]. A logistic map is a polynomial map. It is often cited as an example of how complex behavior can arise from a very simple nonlinear dynamical equation [Ott02]. This map is defined by

$$x_{n+1} = \mu\, x_n (1 - x_n) \qquad (2)$$

Obviously, $x_n \in [0,1]$ under the conditions that the initial $x_0 \in [0,1]$, where n is the iteration number and $\mu = 4$.

3.2. Circle map

The circle map is represented by

$$x_{n+1} = x_n + d - (c/2\pi)\sin(2\pi z_n) \bmod(1) \tag{3}$$

where $c = 0.5$, $d = 0.2$, and $x_0 \in [0,1]$ generates chaotic sequence in [0,1].

3.3. Tent map

It resembles the logistic map. It generates chaotic sequences in (0,1) assuming the following form:

$$X_{n+1} = \begin{cases} X_n/0.7, & X_n < 0.7, \\ 10/3X_n(1-X_n), & \text{otherwise.} \end{cases} \tag{4}$$

3.4. Sinusoidal map

It is represented by

$$X_{n+1} = ax_n^2 \sin(\pi x_n). \tag{5}$$

when $a = 2.3$ and $x_0 = 0.7$ it has the simplified form represented by

$$X_{n+1} = \sin(\pi x_n). \tag{6}$$

It generates chaotic sequence in (0,1).

3.5. Sinus map

Sinus map is defined as follows:

$$X_{n+1} = 2.3(X_n)^{2\ \sin(\pi X_n)}. \tag{7}$$

4. DIFFERENTIAL EVOLUTION

The key idea behind DE is a scheme for generating trial parameter vectors. Mutation and crossover are used to generate new vectors (trial vectors), and selection then determines which of the vectors will survive the next generation [BM07]. A set of D optimization parameters is called an individual, which is represented by a D-dimensional parameter vector. A population consists of NP parameter vectors $X_{i,G}$, ($i = 1,2,...,NP$ for each generation G). According to Storn and Price, DE's basic strategy can be described as follows.

4.1. Mutation

For each target vector $X_{i,G}$ ($i = 1,2,...,NP$), a mutant vector $V_{i,G+1}$ is generated according to

$$V_{i,G+1} = X_{r1,G} + F \cdot (X_{r2,G} - X_{r3,G}), \quad r1 \neq r2 \neq r3 \neq i. \tag{8}$$

where $r1$, $r2$, $r3$ belong to $\{1, 2,...,NP\}$ are randomly chosen integer indexes. Note that indexes have to be different from each other and from the running index. F is called mutation factor between [0,1] which controls the amplification of the differential variation ($X_{r2,G}$ - $X_{r3,G}$).

4.2. Crossover

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector $U_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1})$, that is

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } \mathrm{rand}(j) \leq CR \text{ or } j = rnb(i) \\ x_{ji,G} & \text{otherwise} \end{cases} \quad j = 1,2,\ldots,D. \tag{9}$$

where rand (j) is the jth evaluation of a uniform random number generator between [0,1]. CR is the crossover constant between [0,1] which has to be determined by the user. $rnb(i)$ is a randomly chosen index from $1,2,\ldots,D$ which ensures that $U_{i,G+1}$ gets at least one parameter from $V_{i,G+1}$. Otherwise, no new parent vector would be produced and the population would not alter.

4.3. Selection

To decide whether or not it should become a member of the next generation $G + 1$, the trial vector $U_{i,G+1}$ is compared to the target vector $X_{i,G}$. Assume that the objective function is to be minimized, according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if } f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \tag{10}$$

That is, if vector $U_{i,G+1}$ yields a better evaluation function value than $X_{i,G}$, then $X_{i,G+1}$ is set to $U_{i,G+1}$; otherwise, the old value $X_{i,G}$ is retained. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

5. CHAOTIC DIFFERENTIAL EVOLUTION ALGORITHMS

In the proposed algorithms, when a random number is needed by the classical DE algorithm, it is generated by iterating one step of the chosen chaotic map that has been started from a random initial condition at the first iteration. Also, both constants *F* and *CR* are updated chaotically as shown follows:

$$F_{G+1} = f(F_G),\ \ F_{\min} \le F_G \le F_{\max} \qquad (11)$$

$$CR_{G+1} = f(CR_G),\ \ CR_{\min} \le CR_G \le CR_{\max},\ \ G = 1,2,\ldots \qquad (12)$$

where the values of $F_{\min}$, $F_{\max}$, $CR_{\min}$, and $CR_{\max}$ are user defined.
Along the rest of this section, the proposed Chaotic Differential Evolution (CDE) algorithms will be illustrated as follows

5.1. CDE1

Initial population is generated by iterating the selected chaotic maps until reaching *NP*. In this algorithm, *F* and *CR* values are fixed like in original DE algorithm.

5.2. CDE2

In this algorithm, *F* value has been modified by the selected chaotic maps as in Eq.(11).

5.3. CDE3

In this algorithm, *CR* value has been modified by the selected chaotic maps as in Eq.(12).

5.4. CDE4

In this algorithm, *F* and *CR* values have been modified by the selected chaotic maps as in Eq.(11) and Eq.(12), respectively.

5.5. CDE5

In this algorithm, CDE1 and CDE2 are combined, that is initial population is generated by iterating the selected chaotic maps and *F* value has been modified by the selected chaotic maps when needed.

5.6. CDE6

In this algorithm, that is initial population is generated by iterating the selected chaotic maps and *CR* value has been modified cha chaotically.

5.7. CDE7

CDE1, CDE2, and CDE3 are combined. In this approach, initial population is generated by iterating the selected chaotic maps. *F* and *CR* values have been modified by the selected chaotic maps.

6. EXPERIMENTS AND RESULTS

The initial DE parameters are set at $NP = 50$, $G = 500$ for all proposed algorithms. The value of *F* is set at $F = 0.5$ for algorithms CDE1, CDE3, and CDE6, while $CR = 0.8$ for algorithms CDE1, CDE2, and CDE5. For the rest algorithms $CR_{\min} = 0.7$, $CR_{\max} = 0.9$, $F_{\min} = 0.3$, and $F_{\max} = 0.6$.
The two selected benchmark functions are Griewangk and Rastrigin functions defined in Eq.(13) and Eq.(14), respectively. Table 1 shows the main properties of the selected benchmark functions used in the experiments, where *lb* indicates lower bound, *ub* indicates upper bound, and *opt* indicates optimal point.

$$f_1(x) = \sum_{i=1}^{N}\left(\frac{x_i^2}{4000}\right) - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (13)$$

$$f_2(x) = 10 \times N + \sum_{i=1}^{N}(x_i^2 - 10\cdot\cos(2\pi x_i)) \qquad (14)$$

Table 1: Selected benchmark functions

No.	Function	*lb*	*ub*	*opt.*	property
1	Griewangk	-50	50	0	Multi-modal
2	Rastrigin	-5.12	5.12	0	Multi-modal

Each algorithm was run for 100 times for each function to catch its stochastic properties. The goal is not to find the global optimum values but to find out the potential of the proposed algorithm. Algorithm success rate defined in Eq.(15) has been used for comparison of the results obtained from both functions.

$$S = 100\frac{NT_{successful}}{NT_{all}}|Q_{level} \qquad (15)$$

where $NT_{successful}$ is the number of trials, which found the solution on the Q_{level} in the allowable maximum iteration. NT_{all} is the number of all trials. Q_{level} is the end condition to stop the algorithm, when it converges into Q_{level} tolerance [Ala10]. Tables 2 and 3 depict the success rates of the proposed algorithms for both functions Griewangk and Rastrigin ($N = 2$), respectively.

Table 2: Success rates of CDE algorithms for Griewangk Function (*N* = 2)

Q_{level}	CDE1	CDE2	CDE3	CDE4	CDE5	CDE6	CDE7
Logistic							
1.e-5	45	40	60	70	45	94	53
1.e-6	11	11	19	48	18	62	37
Circle							
1.e-5	47	68	55	62	77	67	82
1.e-6	13	29	18	29	23	39	44
Tent							
1.e-5	46	56	57	50	49	60	83
1.e-6	12	28	17	12	24	18	46
Sinusoidal							
1.e-5	44	48	83	50	54	91	51
1.e-6	12	25	26	14	23	72	13
Sinus							
1.e-5	55	72	75	52	60	49	56
1.e-6	15	29	44	48	27	26	40

Table 3: Success rates of CDE algorithms for Rastrigin Function (*N* = 2)

Q_{level}	CDE1	CDE2	CDE3	CDE4	CDE5	CDE6	CDE7
Logistic							
1.e-5	45	40	55	70	55	93	52
1.e-6	15	10	15	45	11	43	36
Circle							
1.e-5	40	60	45	49	67	58	80
1.e-6	12	25	10	28	22	30	38
Tent							
1.e-5	35	40	35	45	37	42	70
1.e-6	10	27	14	15	27	19	50
Sinusoidal							
1.e-5	40	55	80	55	50	90	57
1.e-6	17	25	24	16	30	77	15
Sinus							
1.e-5	43	70	75	59	61	49	57
1.e-6	17	28	45	41	23	27	32

CONCLUSION

In this paper, Several new chaotic DE algorithms have been proposed and different chaotic maps have been analyzed in the benchmark functions. Five chaotic maps have been embedded to adapt the parameters of the standalone algorithm. This has been done using the chaotic number generators each time a random number is needed by the classical DE algorithm. It has been also shown that, these methods especially CDE4 and CDE6 algorithms have some what increased the solution quality, which gives an indication that the more involvement of chaos in updating the algorithm parameters the better solutions attained.

REFERENCES

[Ala10] **B. Alatas** - *Chaotic harmony search algorithms*, Applied Mathematics and Computation, 216: 2687-2699, 2010.

[BM07] **B. Babu, S. Munawar** - *Differential evolution strategies for optimal design of shell-and-tube heat exchangers*, Chem Eng Sci, 62(14):3720-3739, 2007.

[EA11] **M. F. El-Santawy, A. N. Ahmed** - *The chaotic є-constraint approach*, Journal of American Science, 7(7): 470-472, Marsland Press, 2011.

[EA12] **M. F. El-Santawy, A. N. Ahmed** - *Chaotic Harmony Search Optimizer for Solving Numerical Integration*, Computing and Information Systems Journal, 16(2), University of the West of Scotland, 2012.

[May76] **R. M. May** - *Simple mathematical models with very complicated dynamics*, Nature, 26: 459-467, 1976.

[Ott02] **E. Ott** - *Chaos in Dynamical Systems*, Cambridge University Press, Cambridge, UK., 2002.

[RK07] **J. M. Reddy, N. D. Kumar** - *Multiobjective differential evolution with application to reservoir system optimization*, J Comput Civil Eng, 21(2):136-146, 2007.

[SP95] **R. Storn, K. Price** - *Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical Rep. No. TR-95-012, International Computer Science Institute, Berkley, 1995.