Chaos Driven Evolutionary Algorithm: a New Approach for Evolutionary Optimization

R.Senkerik, M. Pluhacek, D. Davendra, I. Zelinka, Z. Kominkova Oplatkova

***Abstract*—**This research deals with the initial investigations on the concept of a chaos-driven evolutionary algorithm Differential evolution. This paper is aimed at the embedding of simple two-dimensional chaotic system, which is Lozi map, in the form of chaos pseudo random number generator for Differential Evolution. The chaotic system of interest is the discrete dissipative system. Repeated simulations were performed on standard benchmark Schwefel's test function in higher dimensions. Finally, the obtained results are compared with canonical Differential Evolution.

***Keywords*—**Chaos, Differential evolution, Evolutionary algorithms, Lozi map.

I. Introduction

THESE days the methods based on soft computing such as neural networks, evolutionary algorithms, fuzzy logic, and genetic programming are known as powerful tool for almost any difficult and complex optimization problem. Ant Colony (ACO), Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Self Organizing Migration Algorithm (SOMA) are some of the most potent heuristics available.

Recent studies have shown that Differential Evolution [1] has been used for a number of optimization tasks, [2], [3] has explored DE for combinatorial problems, [4] has hybridized DE whereas [5] - [7] has developed self-adaptive DE variants.

This chapter is aimed at investigating the chaos driven DE. Although a several of papers have been recently focused on the connection of DE and chaotic dynamics either in the form of hybridizing of DE with chaotic searching algorithm [8] or in the form of chaotic mutation factor and dynamically changing weighting and crossover factor in self-adaptive chaos differential evolution (SACDE) [9], the focus of this paper is the embedding of chaotic systems in the form of chaos number generator for DE and its comparison with the canonical DE.

This research is an extension of the work [10] and continuation of the previous successful initial application based experiment with chaos driven DE [11] – [13] with simple test functions in low dimensions.

The primary aim of this work is not to develop a new type of pseudo random number generator (PRNG), which should pass many statistical tests, but to try to use and test the implementation of natural chaotic dynamics into evolutionary algorithm as a chaotic pseudo random number generator (CPRNG).

The chaotic system of interest is the simple discrete dissipative chaotic system. The two-dimensional Lozi map was selected as the chaos pseudo random number generators for DE based on the successful results obtained with DE [11] or PSO algorithm [14].

Firstly, Differential Evolution is explained. The next sections are focused on the used chaotic systems and test function. Results and conclusion follow afterwards.

II. Differential Evolution

DE is a population-based optimization method that works on real-number-coded individuals [15]. A schematic of the canonical DE strategy is given in Fig. 1.

There are essentially five sections to the code depicted in Fig. 1. Section *1* describes the input to the heuristic. D is the size of the problem, G_{max} is the maximum number of generations, NP is the total number of solutions, F is the scaling factor of the solution and CR is the factor for crossover. F and CR together make the internal tuning parameters for the heuristic.

Section *2* in Fig. 1 outlines the initialization of the heuristic. Each solution $x_{i,j,G=0}$ is created randomly between the two bounds $x^{(lo)}$ and $x^{(hi)}$. The parameter j represents the index to the values within the solution and parameter i indexes the solutions within the population. So, to illustrate, $x_{4,2,0}$ represents the fourth value of the second solution at the initial generation.

After initialization, the population is subjected to repeated iterations in section *3*.

Section *4* describes the conversion routines of DE. Initially, three random numbers r_1, r_2, r_3 are selected, unique to each other and to the current indexed solution i in the population in

R.Senkerik, M. Pluhacek and Z. Kominkova Oplatkova are with the Department of Informatics and Artificial Intelligence, Tomas Bata University in Zlin, Nad Stranemi 4511, 760 05 Zlin, Czech Republic, (phone: +420576035189; fax: +420576035279; e-mail: senkerik@fai.utb.cz).

D. Davendra and I. Zelinka are with the Department of Computer Science, VŠB-Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic, (e-mail: ivan.zelinka@vsb.cz).

This work was supported by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089, the project IT4Innovations Centre of Excellence No. CZ.1.05/1.1.00/02.0070, Grant Agency of the Czech Republic: GACR 13-08195S, and by the Development of human resources in research and development of latest soft computing methods and their application in practice project: No. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic; and by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2013/012.

4.1. Henceforth, a new index j_{rand} is selected in the solution. j_{rand} points to the value being modified in the solution as given in *4.2*. In *4.3*, two solutions, $x_{j,r1,G}$ and $x_{j,r2,G}$ are selected through the index r_1 and r_2 and their values subtracted. This value is then multiplied by *F*, the predefined scaling factor. This is added to the value indexed by r_3.

However, this solution is not arbitrarily accepted in the solution. A new random number is generated, and if this random number is less than the value of *CR*, then the new value replaces the old value in the current solution. The fitness of the resulting solution, referred to as a perturbed vector $u_{j,i,G'}$, is then compared with the fitness of $x_{j,i,G}$. If the fitness of $u_{j,i,G}$ is greater than the fitness of $x_{j,i,G'}$, then $x_{j,i,G'}$ is replaced with $u_{j,i,G}$; otherwise, $x_{j,i,G'}$ remains in the population as $x_{j,i,G+1}$. Hence the competition is only between the new *child* solution and its *parent* solution.

Description of used DERand1Bin strategy is presented in (1). Please refer to [15] - [18] for the detailed complete description of all other strategies.

$$u_{i,G+1} = x_{r1,G} + F \cdot \left(x_{r2,G} - x_{r3,G}\right) \tag{1}$$

```
1. Input: $D, G_{\max}, NP \geq 4, F \in (0,1+), CR \in [0,1]$, and initial bounds: $\vec{x}^{(lo)}, \vec{x}^{(hi)}$.
2. Initialize: $\begin{cases} \forall i \leq NP \wedge \forall j \leq D : x_{i,j,G=0} = x_j^{(lo)} + rand_j[0,1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \{1,2,...,NP\},\ j = \{1,2,...,D\},\ G = 0,\ rand_j[0,1] \in [0,1] \end{cases}$
3. While $G < G_{\max}$
    $\forall i \leq NP$
        4. Mutate and recombine:
            4.1  $r_1, r_2, r_3 \in \{1,2,....,NP\}$, randomly selected, except: $r_1 \neq r_2 \neq r_3 \neq i$
            4.2  $j_{rand} \in \{1,2,...,D\}$, randomly selected once each $i$
            4.3  $\forall j \leq D, u_{j,i,G+1} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \quad \text{if } (rand_j[0,1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{cases}$
        5. Select
            $\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{if } f(\vec{u}_{i,G+1}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}$
    $G = G + 1$
```

Fig. 1. Canonical DE Schematic

III. The Concept of Chaos Driven DE

This section contains the description of discrete dissipative chaotic maps, which can be used as the chaotic pseudo random generators for DE as well as the main principle of the ChaosDe concept. In this research, direct output iterations of the chaotic maps were used for the generation of real numbers in the process of crossover based on the user defined *CR* value and for the generation of the integer values used for selection of individuals. The initial concept of embedding chaotic dynamics into the evolutionary algorithms is given in [19].

A. Chaotic pseudo random number generator

The general idea of ChaosDE and CPRNG is to replace the default PRNG with the discrete chaotic map. As the discrete chaotic map is a set of equations with a static start position, we created a random start position of the map, in order to have different start position for different experiments. This random position is initialized with the default PRNG, as a one-off randomizer. Once the start position of the chaotic map has been obtained, the map generates the next sequence using its current position.

The first possible way is during the evolutionary process initialization to generate and store a long data sequence (approx. 50-500 thousands numbers) and keep the pointer to the actual used value in the memory. In case of the using up of the whole sequence, the new one will be generated with the last known value as the new initial one.

The second approach is that the chaotic map is not re-initialized during the experiment and no long data series is stored, thus it is imperative to keep the current state of the map in memory to obtain the new output values.

As two different types of numbers are required in ChaosDE; real and integers, the use of modulo operators is used to obtain values between the specified ranges, as given in the following equations (2) and (3):

$$rndreal = \text{mod}\ (\text{abs}\ (rndChaos)\ , 1.0) \tag{2}$$

$$rndint = \text{mod}\ (\text{abs}\ (rndChaos)\ , 1.0) \times Range + 1 \tag{3}$$

Where *abs* refers to the absolute portion of the chaotic map generated number *rndChaos*, and *mod* is the modulo operator. *Range* specifies the value (inclusive) till where the number is to be scaled.

B. Chaotic systems for CPRNG

Following chaotic systems as given in Table I can be used as the CPRNGs.

TABLE I. POSSIBLE DISCRETE CHAOTIC SYSTEMS AS CPRNG AND PARAMETERS SET UP

Chaotic system	Notation	Parameters values
Burgers Map	$X_{n+1} = aX_n - Y_n^2$ $Y_{n+1} = bY_n + X_n Y_n$	a = 0.75 and b = 1.75
Delayed Logistic	$X_{n+1} = AX_n(1 - Y_n)$ $Y_{n+1} = X_n$	A = 2.27
Dissipative Standard Map	$X_{n+1} = X_n + Y_{n+1} (\mathrm{mod}\, 2\pi)$ $Y_{n+1} = bY_n + k \sin X_n (\mathrm{mod}\, 2\pi)$	b = 0.1 and k = 8.8
Ikeda Map	$X_{n+1} = \gamma + \mu(X_n \cos\phi + Y_n \sin\phi)$ $Y_{n+1} = \mu(X_n \sin\phi + Y_n \cos\phi)$ $\phi = \beta - \alpha/(1 + X_n^2 + Y_n^2)$	α = 6, β = 0.4, γ = 1 and μ = 0.9
Lozi Map	$X_{n+1} = 1 - a\lvert X_n\rvert + bY_n$ $Y_{n+1} = X_n$	a = 1.7 and b = 0.5
Tinkerbelt Map	$X_{n+1} = X_n^2 - Y_n^2 + aX_n + bY_n$ $Y_{n+1} = 2X_n Y_n + cX_n + dY_n$	a = 0.9, b = -0.6, c = 2 and d = 0.5

C. Selected chaotic system - Lozi map

Lozi map is the selected example of chaotic systems. The *x*, *y* plot of the Lozi map is depicted in Fig. 2. The map equations are given in (4). The parameters are: *a* = 1.7 and *b* = 0.5 as suggested in [20]. The chaotic behavior of the Lozi map, represented by the examples of direct output iterations are depicted in Fig. 3 (line-plot) and Fig. 4 (point-plot).

$$\begin{aligned} X_{n+1} &= 1 - a\lvert X_n\rvert + bY_n \\ Y_{n+1} &= X_n \end{aligned} \tag{4}$$

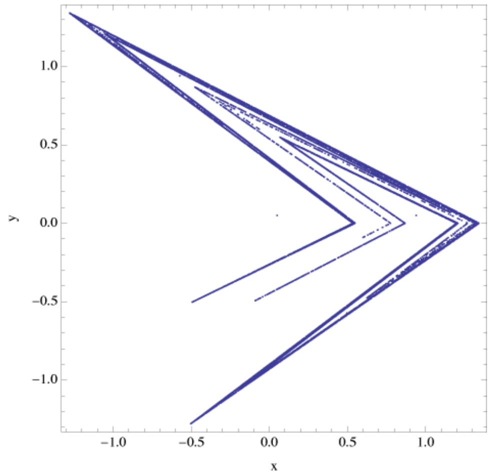

Fig. 2. *x*, *y* plot of the Lozi map

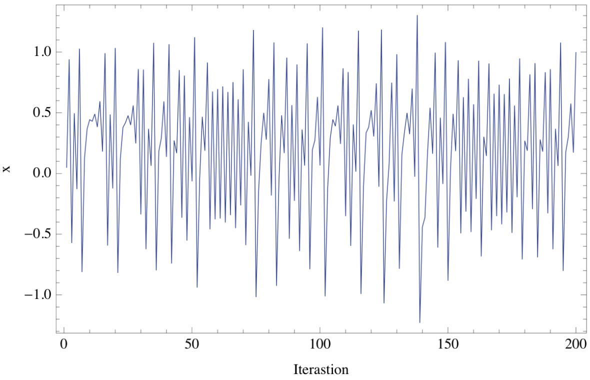

Fig. 3. Iterations of the uncontrolled Lozi map (variable *x*)

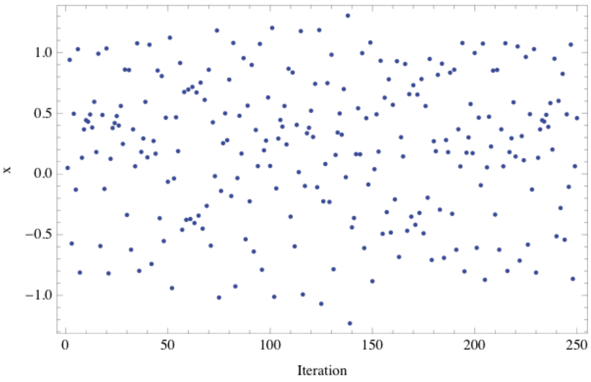

Fig. 4. Iterations of the Lozi map (variable *x* – point-plot)

The illustrative histogram of the distribution of real numbers transferred into the range <0 - 1> generated by means of chaotic Lozi map is in Fig. 5.

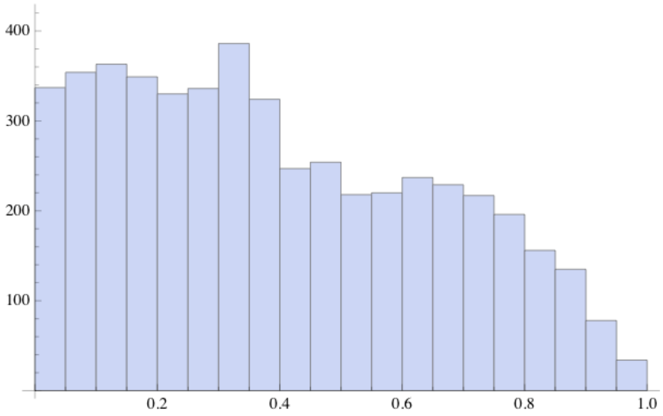

Fig. 5. Histogram of the distribution of real numbers transferred into the range <0 - 1> generated by means of the chaotic Lozi map – 5000 samples

IV. Benchmark Function

For the purpose of evolutionary algorithms performance comparison within this initial research, the Schwefel's test function (5) was selected. The 3D diagram for $D = 2$ is depicted in Fig. 6, and the 2D diagram for $D = 1$ is depicted in Fig. 7.

$$f(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{|x_i|}\right) \tag{5}$$

Function minimum:
Position for E_n: $(x_1, x_2 \ldots x_n) = (420.969, 420.969, \ldots, 420.969)$
Value for E_n: $y = -418.983 \cdot Dimension$

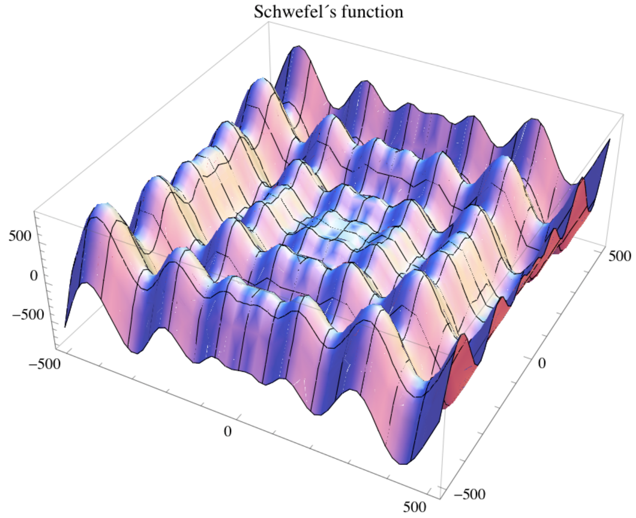

Fig. 6. 3D plot of Schwefel's function

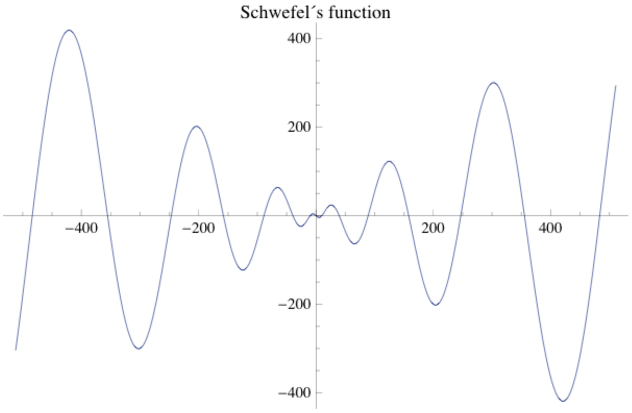

Fig. 7. 2D plot of Schwefel's function

V. Results

The novelty of this approach represents the utilization of discrete chaotic map as a pseudo random number generator for DE. In this paper, the canonical DE strategy DERand1Bin and the Chaos DERand1Bin strategy driven by Lozi map (ChaosDE) were used. The parameter settings for both canonical DE and ChaosDE were obtained analytically based on numerous experiments and simulations (see Table II). Experiments were performed in an environment of *Wolfram Mathematica*, canonical DE therefore used the built-in *Mathematica software* pseudo random number generator. All experiments used different initialization, i.e. different initial population was generated in each run of Canonical or Chaos driven DE.

Within this research, one experiment was performed. It utilizes the maximum number of generations fixed at 3000 generations. This allowed the possibility to analyze the progress of DE within a limited number of generations and cost function evaluations.

The results of the experiment are shown in Table III, which represent the simple statistics for cost function values, e.g. average, median, maximum values, standard deviations and minimum values representing the best individual solution for all 50 repeated runs of canonical DE and ChaosDE.

The main aim of the optimization was to find the global extreme (minimum) of the Schwefel's test function in higher dimensions. For $D = 30$, the global minimum has the following value E_n: $y = -12569.49$.

TABLE II. Parameter set up for canonical DE and ChaosDE

DE Parameter	Value
Popsize	75
F	0.8
Cr	0.8
Dimensions	30
Generations	100·D = 3000
Max Cost Function Evaluations (CFE)	225000

Table IV compares the progress of ChaosDe and Canonical DE. The Table IV contains the average CF values for the generation No. 750, 1500, 2250 and 3000 from all 50 runs.

The bold values within the both Table III and Table IV depict the best obtained results.

TABLE III. SIMPLE RESULTS STATISTICS

CF statistical parameter	Canonical DE	ChaosDE
Average CF	-5944.01	**-10883.5**
Median CF	-5961.63	**-10966.5**
Max. CF	-5412.69	**-7609.94**
Min. CF	-7045.82	**-12427.6**
Std. Dev.	**262.232**	996.605

TABLE IV. COMPARISON OF PROGRESS TOWARDS THE MINIMUM

DE Version	Avg. CF value for Gen. No. 750	Avg. CF value for Gen. No. 1500	Avg. CF value for Gen. No. 2250	Avg. CF value for Gen. No. 3000
Canonical DE	-5341.68	-5591.21	-5831.25	-5944.01
ChaosDE	**-5519.11**	**-7625.59**	**-9501.46**	**-10883.5**

Obtained numerical results and graphical comparisons in Fig. 8 – 11 support the claim that Chaos DE driven by Lozi map has given the best overall results. The graphical comparison of the time evolution of CF values for the best individual solutions (the solution with the minimal final cost function value) for Chaos DE with Lozi map and canonical DERand1Bin strategy is depicted in Fig. 8, whereas Fig. 9 represents the comparison of the time evolution of average CF values from all 50 runs. Fig. 10 shows the time evolution of CF values for the best progressive individual solutions. These individual solutions represent the ones with the lowest sum of the CF values with the step of 20 generations, i.e. with the best progress towards the global optimum. Finally the Fig. 11 confirms the robustness of Chaos DE driven by chaotic Lozi map in finding the best solutions for all 50 runs.

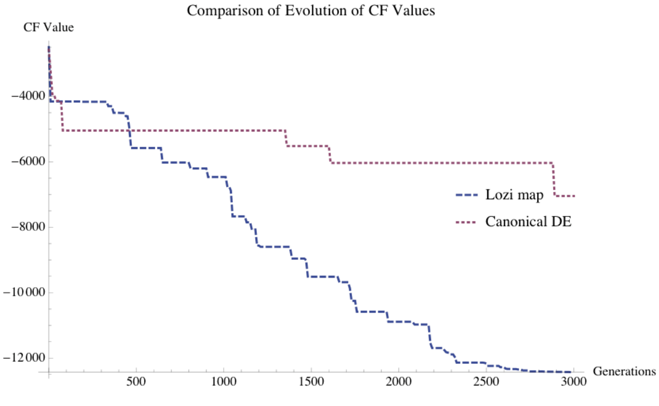

Fig. 8. Comparison of the time evolution of CF values for the best individual solutions, i.e. the solutions with the minimal final cost function value

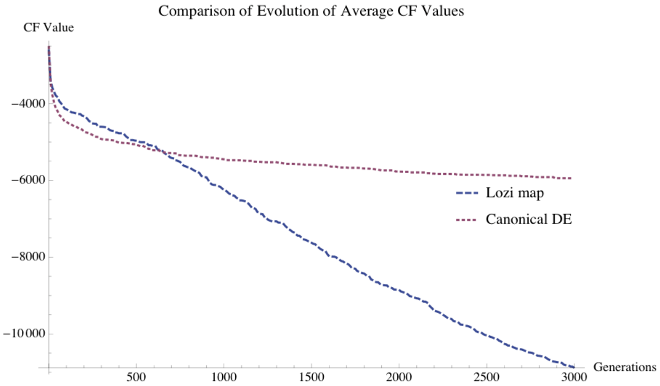

Fig. 9. Comparison of the time evolution of average CF values for all 50 runs of ChaosDE and Canonical DE

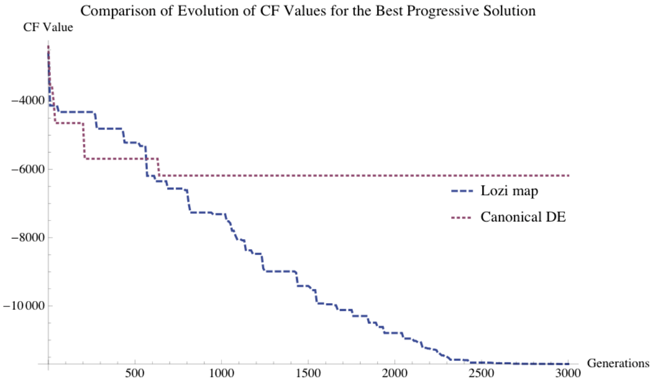

Fig. 10. Comparison of time evolution of CF values for the best progressive individual solutions, i.e. solutions with the lowest sum of the CF values with the step of 20 generations.

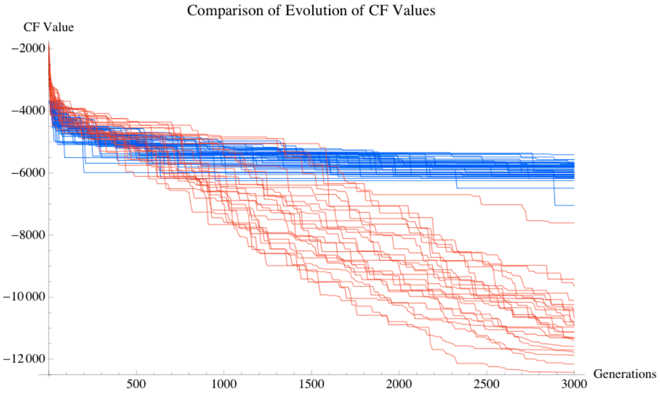

Fig. 11. Comparison of the time evolution of CF values for all 50 runs of canonical DE (blue) and ChaosDE (red)

VI. CONCLUSION

In this paper, chaos driven DERand1Bin strategy was tested and compared with canonical DERand1Bin strategy. Based on obtained results, it may be claimed, that the developed

ChaosDE driven by means of the chaotic Lozi map gives considerably better results than other compared heuristics.

Since this was an initial study, future plans include experiments with benchmark functions in higher dimensions, testing of different chaotic systems and obtaining a large number of results to perform statistical tests.

Furthermore chaotic systems have additional parameters, which can by tuned. This issue opens up the possibility of examining the impact of these parameters to generation of random numbers, and thus influence on the results obtained using differential evolution.

REFERENCES

[1] K. Price," An Introduction to Differential Evolution", In: New Ideas in Optimization, (D. Corne, M. Dorigo and F. Glover, Eds.), McGraw-Hill, London, UK, pp. 79–108, 1999, ISBN 007-709506-5, 1999.

[2] M. F. Tasgetiren, P. N. Suganthan and Q. K. Pan, "An Ensemble of Discrete Differential Evolution Algorithms for Solving the Generalized Traveling Salesman Problem", Applied Mathematics and Computation, Vol. 215, Issue 9, p. 3356 - 3368, 2010.

[3] G. Onwubolu and D. Davendra (Eds), "Differential Evolution: A handbook for Permutation-based Combinatorial Optimization", Springer, Germany, 2009

[4] S. Das, A. Konar, U. K. Chakraborty and A. Abraham, "Differential evolution with a neighborhood based mutation operator: a comparative study", IEEE Transactions on Evolutionary Computations, vol. 13, no. 3, 2009, pp. 526-553.

[5] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization", IEEE Transactions on Evolutionary Computations, vol. 13, no. 2, 2009, pp. 398-417.

[6] J. Zhang and A. C. Sanderson, "JADE: Self-adaptive differential evolution with fast and reliable convergence performance", In Proceedings of IEEE Congress on Evolutionary. Computation., Singapore, 2007, pp. 2251-2258.

[7] J. Zhang and A. C, Sanderson, "Self-adaptive multiobjective differential evolution with direction information provided by archived inferior solutions", In Proceedings of IEEE World Congress on Evolutionary. Computation Hong Kong, 2008, p. 2801 - 2810.

[8] W. Liang, L. Zhang and M. Wang, "The Chaos Differential Evolution Optimization Algorithm and its Application to Support Vector Regression Machine", Journal Of Software, Vol. 6, No. 7, pp. 1297-1304, 2011.

[9] G. Zhenyu, Ch. Bo, Z. Min and C. Binggang, "Self-Adaptive Chaos Differential Evolution", Lecture Notes in Computer Science, 2006, Volume 4221, pp. 972-975, 2006.

[10] R. Senkerik, M. Pluhacek, Z. Kominkova Oplatkova, "Chaos Driven Evolutionary Algorithm: A Novel Approach for Optimization", In: Proceedings of the SCSI 2013, Rhodes Island, GR, EUROPMENT, 2013, pp. 222-226, ISBN 978-1-61804-204-0.

[11] D. Davendra, I. Zelinka and R. Senkerik, "Chaos driven evolutionary algorithms for the task of PID control", Computers & Mathematics with Applications, Vol. 60, Issue 4, 2010, pp 1088-1104, ISSN 0898-1221.

[12] R. Senkerik, D. Davendra, I. Zelinka, M. Pluhacek and Z. Oplatkova, "An Investigation on the Chaos Driven Differential Evolution: An Initial Study", In: Proceedings of 5th International Conference on Bioinspired Optimization Methods and Their Applications, BIOMA 2012, pp. 185 - 194, 2012, ISBN 978-961-264-043-9.

[13] R. Senkerik, D. Davendra, I. Zelinka, M. Pluhacek and Z. Oplatkova, "An Investigation On The Differential Evolution Driven By Selected Discrete Chaotic Systems", In: Proceedings of the 18th International Conference on Soft Computing - MENDEL 2012, 2012, pp. 157 - 162, 2012, ISBN 978-80-214-4540-6.

[14] M. Pluhacek, R. Senkerik, D. Davendra, Z. Kominkova Oplatkova and I. Zelinka, "On the behavior and performance of chaos driven PSO algorithm with inertia weight", *Computers & Mathematics with Applications*, 2013, (article in press), ISSN 0898-1221, DOI 10.1016/j.camwa.2013.01.016.

[15] R. Storn, K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, vol. 11, 1997, pp. 341–359.

[16] K. Price, "An Introduction to Differential Evolution", In: New Ideas in Optimization, (D. Corne, M. Dorigo and F. Glover, Eds.), pp. 79–108, McGraw-Hill, London, UK, ISBN 007-709506-5, 1999.

[17] K. Price and R. Storn, "Differential evolution homepage" (2001) [Online]. Available: http://www.icsi.berkeley.edu/~storn/code.html

[18] K. Price, R. Storn and J. Lampinen, "Differential Evolution - A Practical Approach to Global Optimization", 2005, Springer, ISBN: 3-540-20950-6.

[19] R. Caponetto, L. Fortuna, S. Fazzino and M. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms", IEEE Trans. Evol. Comput, vol. 7:3. 2003, pp. 289-304.

[20] Sprott J. C., "Chaos and Time-Series Analysis", Oxford University Press, 2003.