Information Sciences 587 (2022) 692–719

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Impact of chaotic dynamics on the performance of metaheuristic optimization algorithms: An experimental analysis

Ivan Zelinka [a,*], Quoc Bao Diep [a], Václav Snášel [a], Swagatam Das [b], Giacomo Innocenti [c], Alberto Tesi [c], Fabio Schoen [c], Nikolay V. Kuznetsov [d,e,f]

[a] *Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB-TUO, 17.listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic*
[b] *Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700108, India*
[c] *Dept. Information Engineering (DINFO) – University of Florence, via di Santa Marta 3, Firenze, Italy*
[d] *Faculty of Mathematics and Mechanics, St. Petersburg State University, 198504 Peterhof, St. Petersburg, Russia*
[e] *Faculty of Information Technology, University of Jyväskylä 40014 Jyväskylä, Finland*
[f] *Institute for Problems in Mechanical Engineering of the Russian Academy of Sciences, Bolshoj pr. 61, St. Petersburg, 199178, Russia*

A R T I C L E   I N F O

Article history:
Received 21 June 2021
Received in revised form 25 October 2021
Accepted 30 October 2021
Available online 10 November 2021

Keywords:
Deterministic chaos
Swarm intelligence
Evolutionary algorithms
Algorithm dynamics
Algorithm performance

A B S T R A C T

Random mechanisms including mutations are an internal part of evolutionary algorithms, which are based on the fundamental ideas of Darwin's theory of evolution as well as Mendel's theory of genetic heritage. In this paper, we debate whether pseudo-random processes are needed for evolutionary algorithms or whether deterministic chaos, which is not a random process, can be suitably used instead. Specifically, we compare the performance of 10 evolutionary algorithms driven by chaotic dynamics and pseudo-random number generators using chaotic processes as a comparative study. In this study, the logistic equation is employed for generating periodical sequences of different lengths, which are used in evolutionary algorithms instead of randomness. We suggest that, instead of pseudo-random number generators, a specific class of deterministic processes (based on deterministic chaos) can be used to improve the performance of evolutionary algorithms. Finally, based on our findings, we propose new research questions.

1. Introduction

Chaos (deterministic chaos) is a broad term that refers to dynamical phenomena showing random-like behaviors at a first glance, even if they are generated by deterministic systems. This kind of behavior is typical of processes that are strictly stochastic in nature, such as the motion of molecules in a vessel filled with gas. However since chaotic systems are inherently nonlinear, traditional statistical methods, which are often linear, are inadequate for their study. Chaotic system behaviors share several features with random-like mechanisms, thus making them easily mistaken for random noises.

Deterministic chaos and its applications can be observed in control theory, computer science, physics, biology, and many other fields. Deterministic chaos research can be discussed on two levels. On the first level, it is an object of research where

* Corresponding author.
E-mail addresses: ivan.zelinka@vsb.cz (I. Zelinka), vaclav.snasel@vsb.cz (V. Snášel), swagatam.das@isical.ac.in (S. Das), giacomo.innocenti@unifi.it (G. Innocenti), alberto.tesi@unifi.it (A. Tesi), fabio.schoen@unifi.it (F. Schoen), nkuznetsov239@gmail.com (N.V. Kuznetsov).

https://doi.org/10.1016/j.ins.2021.10.076
0020-0255/© 2021 The Author(s). Published by Elsevier Inc.

the properties of cause and effect are examined, and on the second, it is the use of chaos as a tool. This article focuses on the use of modified chaos in heuristic algorithms to investigate their performance. However, for a better understanding of the issue, we will mention both areas now - chaos as an object of research and chaos as a tool that can be used in other disciplines.

Firstly, we will discuss chaos relating to the phenomenon and object of the research. Chaos was the subject of research in the field of dynamical nonlinear systems as early as the nineteenth century when the well-known French mathematician Henri Poincaré solved the problem of three bodies [1]. The introduction of chaos on the scientific scene did not occur until the 1960s when E.N. Lorenz discovered deterministic chaos in mathematical models modeling weather behavior [2]. From that moment on, chaos was found practically everywhere. Some examples of places chaos can be found are; fluid flow [3], plasma flow and turbulence [4], heart rhythm, ECG and EEG recordings [5–7], biological systems, chemical systems, and economic systems. Around deterministic chaos, a whole mathematical theory has been developed which allows us to grasp this extraordinary phenomenon and work with it. Thanks to this, we can now solve problems that were previously unsolvable and thus develop better technologies. One example of many is the use of wingtip winglets, which divert turbulent flow from the wingtip of away [8]. Due to this and regarding the operation of the aircraft, wings with a smaller area are therefore sufficient, even with less resistance at the same lifting force, thus leading to significant fuel savings. It must also be said that chaos has been observed not only in natural and economic systems but also in man-made technological systems. Most of the time, however, the chaos observed in technological systems was induced by the interaction of human technology with natural systems like atmospheric ones. Surprisingly, chaos has also been discovered in systems such as computer systems and algorithms. For example, chaos is observed in the dynamics of evolutionary algorithms [9,10], i.e. chaos within the algorithmic system. It can be said that chaos is a phenomenon that is not intended to occur only in certain types of systems but a phenomenon that can occur virtually anywhere. Therefore, in modern engineering, it is necessary to consider its presence when designing new technologies. Many such examples can be found in today's scientific literature.

On the second level, chaos is used as a tool in, usually interdisciplinary, research. If we look at deterministic chaos as a tool that can be used in research in other scientific fields, then we also come across a vibrant set of possible applications of deterministic chaos. For example, it can be used in the design of various engineering devices, data encryption, and information transmission, but also in improving the functionality of some types of algorithms. Here we will be focusing on the mutual fusion of chaos and computer sciences with attention to the heuristic algorithm.

Concerning chaos and computer science in general, a lot of interesting applications have been published. In [11], for example, researchers combined deterministic chaos and a pseudo-random number generator. The use of an ultra-weak multidimensional coupling of p 1-dimensional dynamical systems to generate random or pseudo-random numbers is discussed there. In another article, [12] examines the logistic map as a potential pseudo-random number generator and compares it to current pseudo-random number generators. The results of logistic maps are compared to traditional pseudo-random number generation methods. The method is used to calculate the number, delay, and period of the logistic map's orbits with varying degrees of precision. The paper [13] suggests a pseudo-random number generator algorithm, which combines coupled map lattice and chaotic iteration. This algorithm was also put to the test in NIST 800–22 statistical test suits, which are often used in image encryption. The authors of [14] use properties of chaotic systems to construct the CCCBG, a pseudo-random bit generator in which two chaotic systems are cross-coupled with each other. The four basic tests are used to evaluate the bitstreams developed by the CCCBG: mono bit test, serial test, auto-correlation, and Poker test. The NIST suit checks, which are the most rigorous randomness tests, were also used. Paper [15] proposes a binary stream-cipher algorithm using dual one-dimensional chaotic maps, with statistic properties indicating that the sequence is of strong randomness. Similar research is carried out in [16–19].

If we stay focused on chaos and heuristic algorithms, which is the core of this paper, then chaos has previously been found in a variety of processes, including evolutionary systems [9] (the first report on observed chaos inside genetic algorithm dynamics) or [10] which is a book discussing the fusion of chaos and evolutionary algorithms (EAs). Chaos has also been used to substitute pseudo-random number generators (PRNGs) in evolutionary algorithms in recent years. Specifically, the use of chaos inside EAs is reported in [17] (discussing whether or not pure chaotic sequences can improve the performance of evolutionary algorithms). Alteratively, papers [20,21] discuss the use of deterministic chaos inside the particle swarm algorithm instead of PRNGs, [11–15] investigate relations between chaos and randomness or the others like [22] (applies chaotic genetic algorithm on cyclic electric load forecasting), [23] (uses chaos driven differential evolution on optimization of the batch reactor) and [24] which uses chaos driven evolutionary algorithms for PID control. Other papers reporting the use of chaos in algorithms are [25] (use of the chaos-based evolutionary algorithm in nonlinear programming) and [26] which discusses adaptive differential evolution algorithms powered by multi-chaotic framework used for parent selection and tested on CEC2014[1] benchmark set. Furthermore recent algorithms like a Whale algorithm [27] with chaos was reported in [28,29] (using chaos game optimization). Another recent algorithm, the Invasive Weed Optimization algorithm (IWO) [30] based on chaos theory, was used for the optimal design of PID controller is principally similar to [24,31]. Of course, our list of classical and modern algorithms using chaos to increase their performance does not end here. For example, a paper using multiple chaoses embedded gravitational search algorithm, paper [32], is another representative of a new generation of algorithm testing chaos hybridization and the optimization algorithm itself. The list of all research papers that contain a fusion of

[1] http://www5.zzu.edu.cn/cilab/info/1005/1013.htm

chaos and a heuristic algorithm would be for the article itself is not the goal of this work, and therefore we refer only to this brief list.

Based on those and other research papers, it can be stated that several interesting contributions have been made dealing with the use of chaos within optimization algorithms. It is worth mentioning work [33], which also deals with the chaos use in a newly designed algorithm, which has been tested on a large number of test functions. Additionally, [34] investigates the performance, scalability and convergence and robustness of chaos-enhanced evolutionary algorithms with boundary constraints while papers like [35–37] or [38] deal with chaos application inside evolutionary algorithms. Therefore, it is clear that recently the interest of scientists has become to be attracted by the hybridization of modern optimization algorithms and chaotic dynamics. The mentioned publications uses chaos application within algorithms from different points of view, but primarily only as an improvement of one specific algorithm, tested for significant test functions. Some of them compared the impact of a chaotic version on a CEC (Congress of Evolutionary Algorithms) test benchmark function (however older and typically one set) to a non-chaotic version of the same algorithm.

We have agreed to broaden the scope of this research to include a newer CEC collection of benchmark test functions, as well as more algorithms of various types (evolutionary vs swarm). We also included one critical aspect of the experiment: we looked at the effect of *modified chaos* on algorithm performance. This paper focuses on using deterministic chaos to produce N periodic sequences (forced by low calculation precision), which are used in evolutionary algorithms instead of pseudo-random number generators or just pure deterministic chaos series. Various EAs of different kinds are used here, see Table 3. Algorithms itself did not performed any analysis on how and if used pseudo-random numbers are random, what is its distribution etc. Pseudo-random, chaotic, and periodic (chaos-based) series are **only simply used**.

A research study, especially a theoretic one, dealing with how much the performance of different algorithms depends on different levels of chaotic dynamics has not yet been satisfactorily developed. For these reasons, this work was created, which aims to point out how much the performance of the evolutionary algorithm depends on the specifically modified chaotic dynamics, which is used inside the algorithm instead of a pseudo-random number generator. This research also raises interesting research questions that suggest an interpretation of evolutionary algorithms on the level of discrete dynamic systems with feedback and thus allow the use of the theoretical mathematical apparatus of cybernetics to analyze and describe it. This could explain why chaos has a positive effect on the performance of evolutionary algorithms. Applying this existing mathematical apparatus to evolutionary and swarm intelligence algorithms shall help shed light on the many unanswered questions in the algorithm community today while providing exciting and powerful mathematical tools for researching these algorithms. However, answering this question belongs to future research. EAs with deterministic chaos systems (DCHS), as demonstrated here, produce mostly the same or better results, as seen in the section *Results*.

The article's structure is as follows. We will clarify the relevance of research in this direction and point out the unanswered questions in the use of chaos in evolutionary algorithms in the *Motivation and Novelty* section. The main idea of our paper is then repeatedly formulated in the *Hypothesis* section, where we repeat the basic ideas from the field of various heuristic algorithms using deterministic chaos and explain why we chose the procedure used in this paper. The *Experiment design* section follows, which explains the conditions under which our experiments were conducted including a link to GitHub with all source codes from our experimentation. We also explore and discuss the impact of accuracy on the chaotic course's periodicity, as well as the nature of our own experiments in the section *Results*. The CEC test functions were used to test ten selected evolutionary algorithms. Whole idea is captured in Fig. 9. The results are discussed and summarized in the section *Conclusions*.

2. Motivation and novelty

The experiments described in this article were motivated by the current state, which was discussed in the previous section, as well as the widespread perception that operations like mutations and others that need random operations cannot be performed without randomness. From the description of evolutionary algorithms, it is evident, that they are discrete dynamic systems. That means they are feedback systems processing their output (in this case the population with fitness) as the input for the next generation procedures. Which individual will be used in the following generation (iteration, migration, etc.) is determined by an objective function (e.g. fitness). If we consider evolutionary or swarm algorithms to be dynamic systems, the assumption that successful control of these systems requires (pseudo) random processes is a bit odd in regards to what the control engineering community normally expects (remember that in control theory randomness **usually** represents a signal, that has to be eliminated).

As a result, our tests were meant to see if the pseudo-random numbers created by the traditional type of pseudo-random number generators (Mersenne - Twister like) are actually necessary for the performance of evolutionary algorithms, or if alternative processes, such as those generated by deterministic chaos, can be used instead. It is also possible to create periodic series (from chaotic ones) of various lengths. Chaos as well as pure randomness, does not exist in computers, only pseudo-chaotic or pseudo-random processes are present. Chaos, unlike pseudo-random processes generated by other algorithms, may alter the length of its period by altering the precision of its calculation. For example, if we employ a logistic Eq. 1 with the parameter $A = 4$ and let this equation loop in a computer, we have a theoretically quasi-random series (for the external observer) that never repeats and, in fact, belongs to the world of deterministic chaos due to the high accuracy in which today's computers work. When the precision of the calculation is reduced, however, rounding happens, resulting in the effect that this otherwise potentially endlessly long series of numbers begins to be shorter and recurrent. As shown in Fig. 2–8, the occurrence of repetitions is dependent on the extent of the computation accuracy.

This study's tests are not only based on determining the impact of chaotic dynamics as pseudo-random numbers on the algorithm's performance, but also on **determining if modifying the computation accuracy or shortening the chaotic period will alter the EAs' performance quality**. That means no pure chaotic series were used in EAs instead of randomness. We employed periodic series of various lengths N based on the chaotic generator "truncating", which has never been done before. It is also debated whether pseudo-random processes are required or whether short pseudo-chaotic processes generated using a deterministic method are sufficient. This raises several new research problems, such as whether EAs require pseudo-random processes to work, and whether there is a link between the type of test function (which typically contains nonlinear elements and it can be seen as a nonlinear system interacting with a specific algorithm) and algorithm performance. All of this is demonstrated by our findings. In the conclusion section, further possible questions and research ideas are presented in the section *Open Questions and Future Research*. Therefore, if we can summarize the previous information, it can be stated that the novelty of our contribution lies, in comparison with others, in the following facts. Our experiments consider not only one or two algorithms but a total of ten which reasonably represent the whole spectrum of evolutionary algorithms and swarm intelligence. Furthermore, these algorithms are applied to recognized benchmark test functions and evaluated by a widely accepted statistical validity test. Another novelty of this paper is that in addition to the influence of pure chaos on the performance of the algorithm, we also study the influence of periodic behavior generated by a given chaotic system. These series are no longer random-like or chaotic but fully deterministic. As our experiments have found that, from a certain length of the N periodic behavior, the algorithms usually start to show better performance concerning randomness or pure chaos. This raises many interesting questions about the dynamics and performance of evolutionary algorithms as such.

Regarding the used chaotic system, we chose the logistic equation, which is a simple representative of chaos from the area of chaotic dynamics. It is very well known, well studied, and sufficiently rich for our experiments. Of course, other chaotic systems can be used, of which there are many today [39]. However, given our experience with these systems and evolutionary techniques, we believe that the logistic equation is quite suitable for the set of experiments and confirmation of our hypotheses presented in this article. Especially if we *force* the chaos generator to produce N periodic behavior. The use of other chaotic systems and other evolutionary algorithms is certainly an issue to be investigated in future researches.

3. Hypothesis

As mentioned in previous sections, a large number of research papers have been produced in the last ten to fifteen years discussing the presence of chaos in evolutionary algorithms, the use of these algorithms for synthesis and chaos control and also the use of chaos as a random generator in the algorithms themselves. If we think about this development, then it is clear that algorithms using chaos more or less benefit in terms of increasing the performance of evolutionary algorithms, which is evident in already published works. If the first generation of researchers used EAs with pseudo-random generators, these EAs were improved by chaos generators; then the question arises as to whether this research could be further extended in the same direction. High-quality random generators produce time series that really repeat after a very large number of iterations, similarly to chaos. Oppositely, chaos as a parametrically adjustable system can change the degree of chaos, which turns into deterministic windows, which can be called N periodic orbit-series.

Our first idea was to use these deterministic windows, Fig. 1, to have N periodic orbits and thus replace the random series produced by pseudo-random number generators or chaotic generators. The problem is that the number of deterministic windows in each chaotic system is limited and usually of very low periodicity (e.g. $N = 4, 8, \ldots$), see Fig. 1. We decided to work around this with a simple trick, i.e. the low accuracy of the calculation, which results in the respective chaotic generator producing periodic series. In other words, the limited accuracy of the calculation degrades the chaos generator to a generator of periodic events (e.g. Fig. 2–8).

Thus a question of whether a chaotic generator operating in forced deterministic regime mode has the same or better benefit of an evolutionary algorithm performance exists. To avoid using only a few existing deterministic windows (existing in virtually every chaotic system), we decided to force the generator to create periodic series by changing the accuracy of the calculation. In other words, based on the accuracy of the calculation of the individual iterations, which then enter back into the chaotic generator, we are able to force the generator to generate periodic series. Furthermore, these are then used as a source instead of a random number generator. Thus, we got into an area that has not yet been explored and deals with the question of whether short or long periodic series can be used in evolutionary algorithms instead of a random number generator. Of course, we could do without chaotic number generators and immediately generate some pre-selected N periodic series. However, it makes sense to use already proven chaotic systems that can be relatively easily forced to generate periodic series. There is such a logical continuity: chaos replaced randomness, periodicity replaced chaos.

The whole principle is shown in Table 1. We, therefore, formulated a hypothesis: *What is the effect of using periodic series instead of a random number generator in evolutionary algorithms? Is there any impact, positive or negative?*

The findings and our answers then open up many other interesting research questions, which seem to connect these algorithms with the theory of discrete dynamical systems and thus allow new insights into evolutionary dynamics.

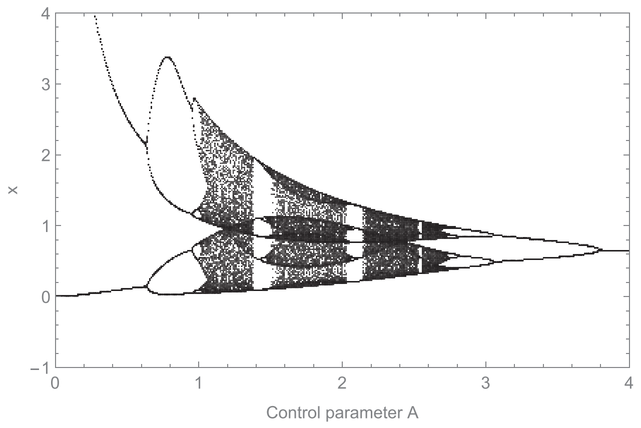

Fig. 1. Chaotic system behavior with deterministic windows containing N periodic behavior. The grainy area is chaos, the deterministic windows are for example at $A = \{1.4, 2.1, 2.55\}$, see [10].

Table 1
Hypothesis and selected literature representing individual periods of development and use of evolutionary algorithms.

Classic era of evolutionary computation using different pseudo-random number generators.	Evolutionary computation using deterministic chaos instead of pseudo-random number generators.	Can periodic series generated by deterministic chaos systems, used instead of pseudo-random number generators, improve evolutionary algorithms performance?
[40–52], . . .,	[53–56] [57], . . .	[58,59], this paper

4. Experiment design

The experiments that have been performed can be divided into two categories. The first category examines how the existence of periodicity generated by deterministic chaos systems is influenced by computation precision, while the second one employs periodical time series generated by chaotic systems inside EAs and compared with the same EAs powered by PRNGs.

Our experiments were designed according to the following facts. To verify/demonstrate our hypothesis about chaos usability it must be firstly remembered that there are a large number of algorithms, test problems, and repeated simulations with different initial conditions (necessary to demonstrate the robustness of the results). All this must then be statistically evaluated, and it is clear that it is not possible to cover all possible scenarios. Therefore, we focused on a narrower selection of more well-known and less well-known algorithms, both in the field of classical evolutionary algorithms and in the field of swarm intelligence. These are algorithms such as differential evolution (DE) of particles swarm (PSO), grey wolf algorithm (GWA), SOMA algorithm, and many others. These algorithms were tested regularly on selected test functions used at CEC. Regular statistical tests and evaluations were then performed on a huge amount of results, which in the end allowed us to evaluate the impact of using chaos with different degrees of accuracy on the performance of these algorithms.

The experiments were performed on a standard PC with a classic i7 processor and 8 GB of memory, as well as partly on a supercomputer of the national supercomputer center IT4 (https://www.it4i.cz/en) to speed up the calculations and shorten the time. There was no other reason to use a supercomputer - these experiments can be repeated on any fast enough computer. Only the logistic Eq. 1 was used to generate chaos because it is a well-known basic system for generating chaos. Our study can be extended by a larger class of chaotic systems. However, it is clear from the results we obtained that extending a larger class of chaotic systems would only yield more simulations, but the results would probably not be fundamentally different.

5. Generators of chaotic or periodic series?

The so-called logistic equation (Eq. 1) was used to demonstrate the reliability of pseudo-chaos periodicity on numerical precision. Several experiments were carried out, and the results are displayed in Fig. 2–8. Other chaotic generators of various mathematical descriptions, such as Lozi, Henon, Ikeda, or others, like those experimentally manufactured and published in [39], can also be employed. On the other hand, we recall that the basic idea of this paper is the use of N periodic series generated by a chaotic generator due to the reduced accuracy of the calculation. Thus, our work differs from previous ones, where only the influence of purely chaotic generators was investigated (see, e.g. [53–57] amongst the others).

In chaotic dynamics it can be also observed the existence of pseudo-patterns, that is, the situation where a time series does not repeat itself exactly, but it periodically displays a very similar behavior. This is a feature of true chaos, which has an inner structure made of a ”basic recurring shape”, which is persistently approached by the system trajectory. Each pseudo-pattern has its unique features, which makes a number of the sequences to be somewhat correlated with the previous ones. As a consequence, the probability density function is no longer evenly distributed among different sequences of numbers, which is a strong difference from white noises. Therefore, using chaos for random-like numbers can make the generator ”biased”, as well as the algorithms which exploit it. However, to observe this, shorter parts of “clear” chaotic series shall be taken into consideration where these pseudo-patterns are dominant (i.e. take a major part of chaotic series). In our experimentation, we have used max 100 000 cost function evaluations, which means more chaotic numbers have been used for one algorithm run (remember, a ”randomness” is generally needed more than once, to build new offspring for one cost function evaluation). Thus, the eventual occurrence of pseudo-patterns should have a negligible impact on algorithm performance. Due to the divergence of nearby trajectories, such patterns shall shortly end if appears in chaotic series.

$$x_{n+1} = Ax_n(1 - x_n) \tag{1}$$

The logistic equation has been employed in several stages of research here as well as in another papers, i.e.

- Identification of a N periodic orbit with numerical precision as a criterion, see Fig. 2–8.
- A global picture of the behavior of a logistic equation with a fixed parameter A = 4, see Fig. 3 and 4.
- A broad view of the behavior of logistic equations considering a variety of parameters $A \in [3.4, 4]$ and numerical precision $\in [1, 20]$, logistic equation has been iterated for 1 000, 10 000, 100 000 iterations, see Table 2.
- Investigate and visualize why and when N periodic series occur, see Fig. 3–4.
- The analysis of the dynamics of logistic equation has been done. For parameter $A = 4$, numerical precision $\in [1, 13]$, initial conditions $x_{start} \in [0.01, 0.99]$ (x was incremented periodically by = 0.01) and 1 000 000 iterations for each combination of this parameters, see Table 2. see Fig. 5–7,.

The impact of the precision on the dynamics of logistic equation is depicted on Fig. 3 (compare with Fig. 4).The original mapping function is changed to a step-wise mapping function with low accuracy. This is the origin of many of the periodic series, that we later used in our studies.

5.1. Determinism or randomness

If we are talking about using random or, more precisely, pseudo-random generators, it is necessary to realize the difference between a purely random process and a pseudo-random process generated by an algorithm, i.e. the difference between true random and pseudo-random numbers. In principle, there are two ways to generate random numbers. The first one is to generate so-called pseudo-random numbers (PRNG), which consists of producing seemingly random numbers via a suitable algorithm. The second way is to generate random numbers using the true random number generator (TRNG), which uses randomness obtained from physical phenomena. Recognition of true and pseudo-random numbers can be very problematic. Statistical analyses are used to verify the trustworthiness of algorithms.

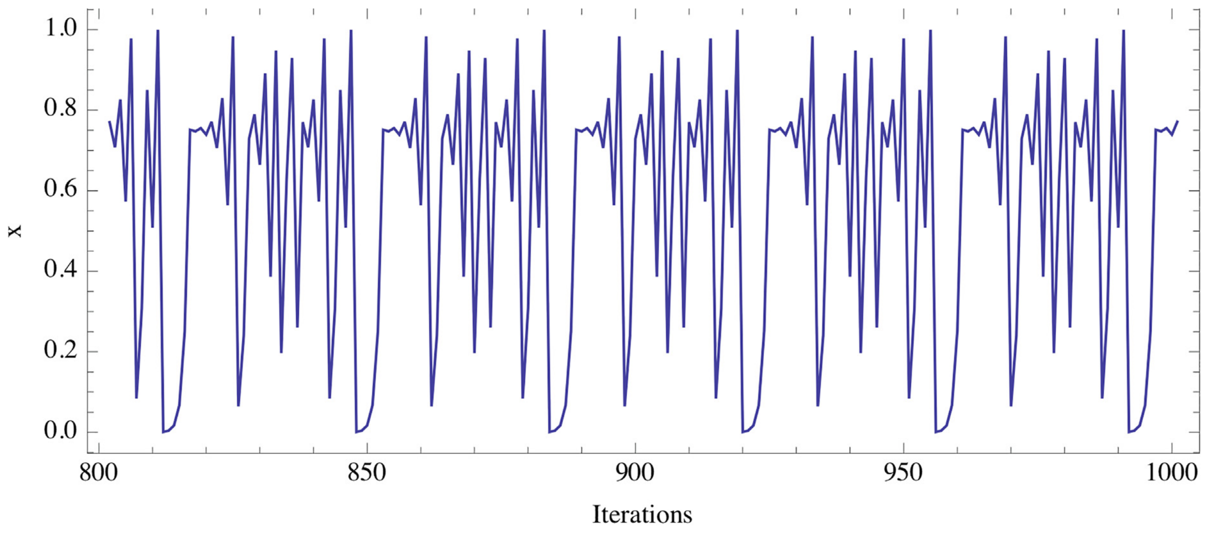

Fig. 2. The period 36 (precision = 4) based on Eq. 1 for $A = 4$, see Table 2.

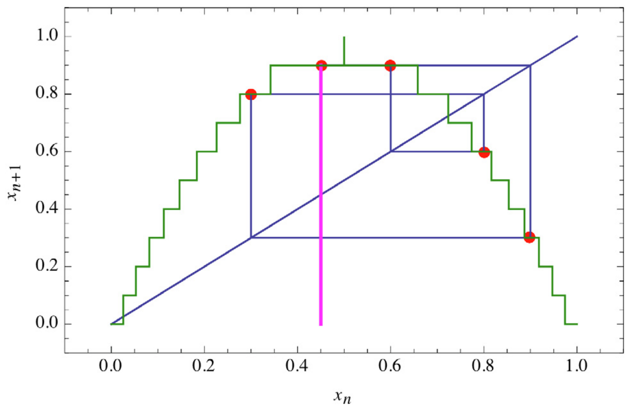

Fig. 3. Cobweb diagram: a low numerical precision = 1 (one decimal behind zero) impact on the mapping function shape. Four periodic orbit is observable (blue line).

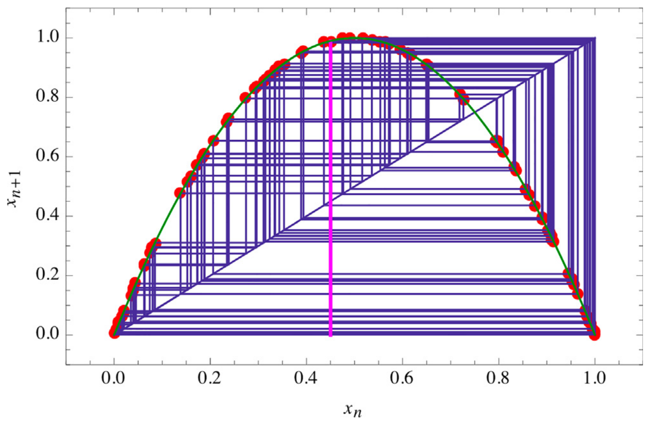

Fig. 4. Cobweb diagram: full chaotic dynamics with of logistic equation. Compare with Fig. 3, the same initial conditions are used.

Table 2
Periodicity dependence of Eq. 1 on various numerical precision (up to precision 13).

Numerical Precision	Minimal Period	Maximal Period
1	4	4
2	2	10
3	10	29
4	15	36
5	67	170
6	143	481
7	421	758
8	1030	4514
9	2277	11227
10	2948	35200
11	9668	57639
12	65837	489154
13	518694	518694

TRNG generation methods use physical/hardware units. A physical random number generator can be based on many principles, one of which may be the unpredictability of the random behavior of atomic and subatomic phenomena, which can be observed using the laws of quantum mechanics. One of them is the use of a radioactive source. The time intervals in which the radioactive source decays are entirely unpredictable. Another interesting feature was the Lavarand generator, built by Silicon Graphics and using lava lamp images to generate truly random numbers. It no longer works today. Closer to common options are devices that use network peaks, measuring noise or user input, such as mouse movement, keystroke delay, and more.

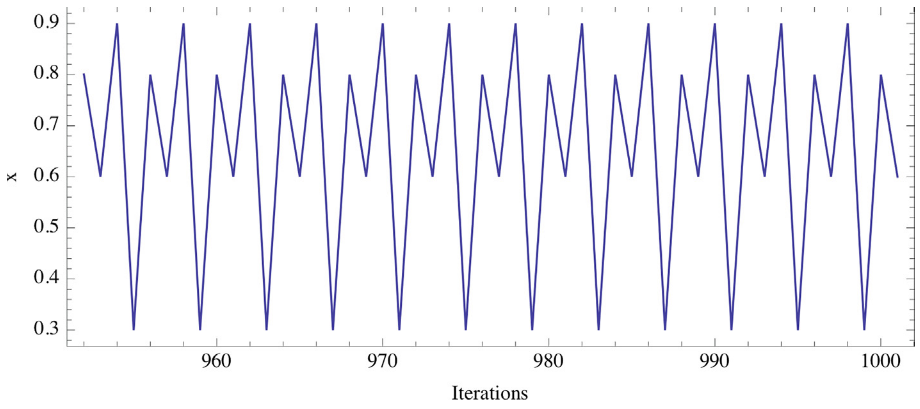

Fig. 5. Periodicity of logistic equation under precision 1 and $x_{start} = 0.45$.

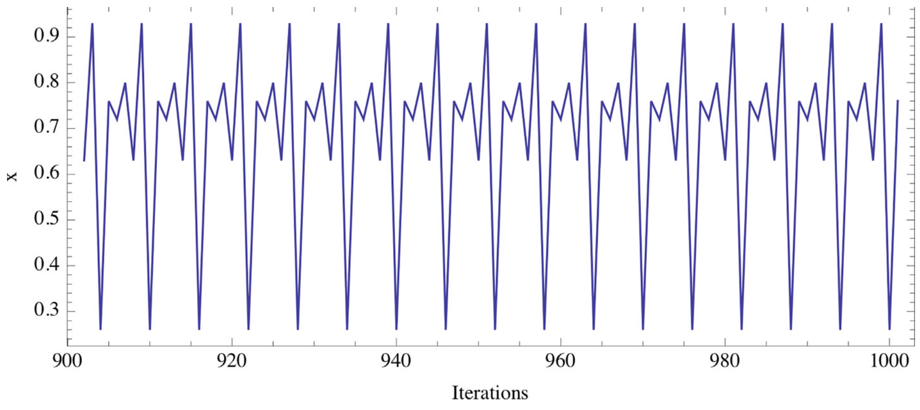

Fig. 6. Periodicity of logistic equation under precision 2 and $x_{start} = 0.8$.

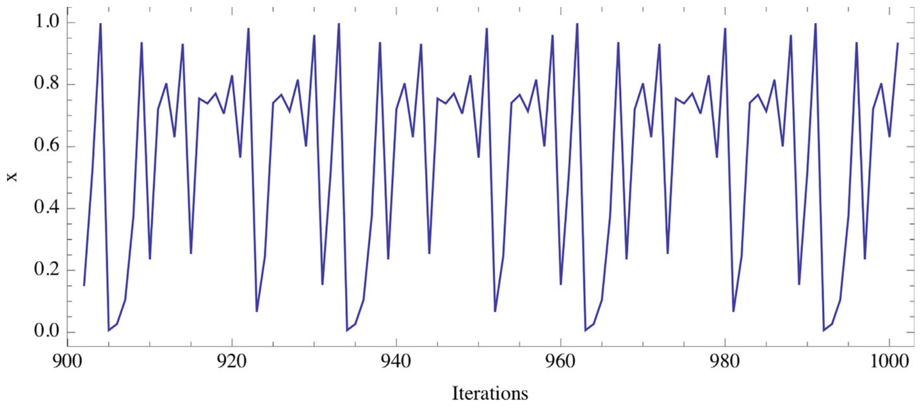

Fig. 7. Periodicity of logistic equation under precision 3 and $x_{start} = 0.6$.

However, computer/software methods are important for our needs. Pseudo-random number generators are algorithms that produce long strings of numbers having a seemingly good random distribution, but later these sequences are repeated, and the quality of the distribution decreases. One of the most used algorithms is a linear congruent generator operating based on the recurrent relationship, Eq. 2. The maximum number of numbers it can create is modulo b$\bmod m$. Slightly different values of the multiplication factor are used to avoid some undesirable properties of the linear congruent generator. Most programming languages have special libraries or functions for creating pseudo-random numbers integrated. Nowadays, of course, they are used in much more modern algorithms. Their output are number series, which for common

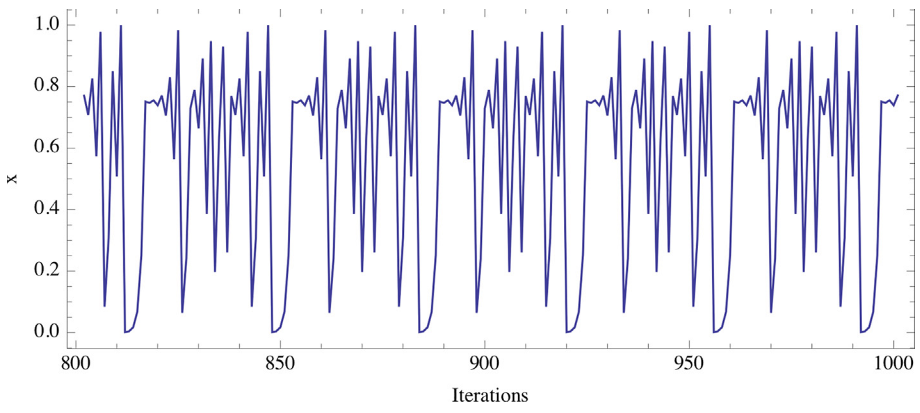

Fig. 8. Periodicity of logistic equation under precision 4 and $x_{start} = 0.02$.

Table 3
The detailed control parameters of the algorithms.

Algorithms	The control parameter values
DE	$pop = 100; F_{min} = 0.4; F_{max} = 1; Cr = 0.5$
SOMA	$pop = 50; Step = 0.11; PRT = 0.1; PathLength = 3.0$
PSO	$pop = 100; w_i = 1; w_{damp} = 0.99; c_1 = 1.5; c_2 = 2.0; v_{max} = 0.1 \times [-100, 100]^D; v_{min} = -v_{max}$
ABC	$pop = 50; n_{onlooker} = pop; l = 0.6D.pop; a = 1$
ACO	$pop = 50; n_{sample} = 40; q = 0.5; \zeta = 1$
FA	$pop = 25; \alpha = 0.2; \beta_0 = 1; \gamma = 1; \alpha_{damp} = 0.98; \delta = 0.05.[-100, 100]^D$
CA	$pop = 50; \alpha = 0.3; \beta = 0.5; p_{accept} = 0.35; n_{accept} = p_{accept}.pop$
GWO	$pop = 30; \vec{A} = 2\vec{a}.\vec{r_1} - \vec{a}; \vec{C} = 2.\vec{r_2}$
WOA	$pop = 30; b = 1; l = (a_2 - 1)rnd + 1$
HFPSO	$pop = 30; \alpha = 0.2; \beta_0 = 2; \gamma = 1; c_1 = c_2 = 1.49445; v_{max} = 0.1 \times [-100, 100]^D; v_{min} = -v_{max}; w_i = 0.9; w_f = 0.5$

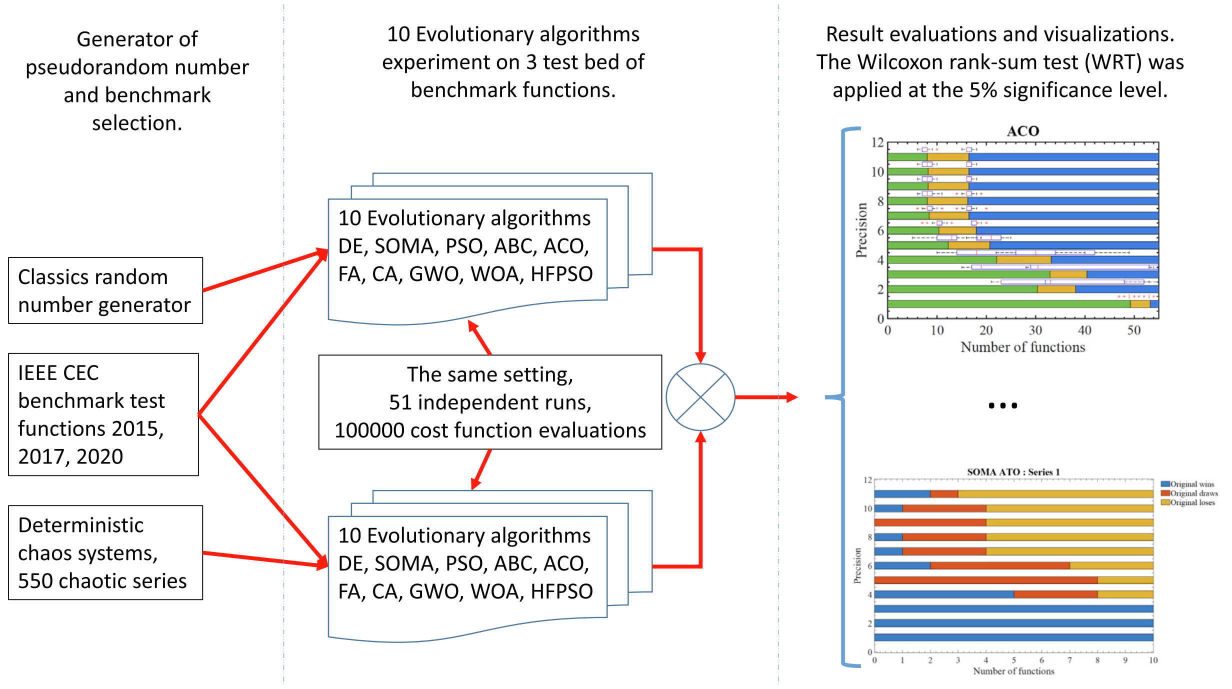
Generator of pseudorandom number and benchmark selection.
10 Evolutionary algorithms experiment on 3 test bed of benchmark functions.
Result evaluations and visualizations. The Wilcoxon rank-sum test (WRT) was applied at the 5% significance level.
Classics random number generator
IEEE CEC benchmark test functions 2015, 2017, 2020
Deterministic chaos systems, 550 chaotic series
10 Evolutionary algorithms DE, SOMA, PSO, ABC, ACO, FA, CA, GWO, WOA, HFPSO
The same setting, 51 independent runs, 100000 cost function evaluations
10 Evolutionary algorithms DE, SOMA, PSO, ABC, ACO, FA, CA, GWO, WOA, HFPSO
ACO
Precision
Number of functions
SOMA ATO : Series 1
Original wins
Original draws
Original loses
Precision
Number of functions

Fig. 9. Experiment idea.

engineering needs behave random-like. One of the most frequently used random number generators is, for example, Mersenne Twister [60].

$$X_{n+1} = (aX_n + b)\text{bmod}m \tag{2}$$

In our paper, we used not only pseudo-random generators, we also employed time series generated by chaos generators (e.g. Fig. 2 and Eq. 1). Due to the fact, that numerical precision has clearly an impact on the occurrence of periodicity in chaos, logistic equation, Eq. 1 has been selected. The data series is generated by this equation (under numerical precision from the interval [1,13]) and with the setting A = 4, Table 2 shows minimal and maximal period observed for the current setting. The repeated generation of those series was started on pseudo-random initial conditions. Algorithms selected for our experiments are reported in Section 6.2. The setting of all versions of algorithms used are referred to in Table 3. It is simple to compute how many times the data series of pseudo-chaotic numbers (PCHNs) generated by DCHS have been used in EAs using this configuration and algorithm architecture.

All PRNG numbers were replaced by PCHNs (with different numerical precision and thus period length). As for comparison, the same algorithms in their canonical versions have been used, with classical PRNGs, just to compare performance.

In our experiments with chaos series, the PRNG has been used only once in each experiment - at the beginning of the chaotic series as the seed for the start of chaotic system iteration. This means if a chaotic series was generated X times, then PRNG was used to uniformly generate *X* pseudo-random numbers in the interval [0,1] that has been used as the initial starting value for the chaotic generator. It is evident that seeding is the most important factor. Certain starting conditions can converge in a finite number of iterations into fixed points, or short period solutions, which are definitely not suited for the scope. They could be referred to as ”pathological traps”. However, excluding such initial seeds would be out of the scope of our experiment use, and hence our experiments demonstrate that using chaos instead of PRNG, is still very robust and such pathological traps have likely rare and negligible impact.

5.2. Experiment idea

Based on the described facts and procedures, our experiments can be summarized as follows. In the first phase, test benchmarks from the IEEE CEC Congress from 2015, 2017, and 2020 were prepared. Classic pseudo-random number generators such as Mersenne Twister and deterministic chaos of the logistic equation type were used to generate purely chaotic series, including periodic series, for a total number of 550. In the following phase, when the experiment itself took place, ten selected evolutionary algorithms were used. For each algorithm, 51 independent runs were performed with a maximum limit of 100,000 evaluations of the cost function. All this was repeated both for the pseudo-random number generator and with the chaotic and periodic series within the already mentioned algorithms. Both phases of the experiment were performed for comparing the performance of the algorithms powered by a pseudo-random number generator or a chaos generator. All these experiments and their results were stored in data files, which were then analyzed and visualized. The experiment principle is shown in the Fig. 9.

6. Computational setup

6.1. Test functions

To rigorously and thoroughly evaluate the impact of a deterministic chaos system on evolutionary algorithm performance, three well-known test beds of the IEEE Congress on Evolutionary Computation (IEEE CEC) are used and listed below.

- The first benchmark set is the IEEE CEC 2015 Competition on Learning-based Real Parameter Single Objective Optimization (CEC 2015, [61]);
- The second is the IEEE CEC 2017 Special Session and Competition on Single Objective Real Parameter Numerical Optimization (CEC 2017, [62]);
- The last one used is the IEEE CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization (CEC 2020, [63]).

The CEC 2015, 2017 and 2020 contain 15, 30 and 10 fitness functions, respectively. **A total of 55 functions** were used, including 6 Unimodal, 10 Simple Multimodal, 3 Basic, 16 Hybrid, and 20 Composition functions, challenging enough to rate the performance of any evolutionary algorithm. All test functions have shifted global optimum and are scalable, see [61–63] for more information.

6.2. Test algorithms

Ten popular algorithms, including the evolutionary computation and swarm intelligence algorithms, were selected for the experiment and listed below.

- Differential Evolution - DE/best/1/bin (DE) [40];
- Self-Organizing Migrating Algorithm AllToOne version (SOMA-ATO) [44,64];
- Particle Swarm Optimization (PSO) [65];
- Artificial Bee Colony Algorithm (ABC) [50];
- Ant Colony Optimization for Continuous Domains (ACO) [41];
- Firefly Algorithms (FA) [66];
- Cultural Algorithm (CA) [67];
- Grey Wolf Optimizer (GWO) [68];
- The Whale Optimization Algorithm (WOA) [42];
- A Hybrid Firefly and Particle Swarm Optimization Algorithm (HFPSO) [43].

These selected algorithms usually employ many types of PRNGs, including uniformly distributed pseudo-random numbers, continuous uniform pseudo-random numbers, normally distributed pseudo-random numbers, uniformly distributed pseudo-random integers, and so on. In our case Mersenne Twister PRNG was used. The performance of the algorithm using PRNGs is compared with its performance when replacing PRNGs with a deterministic chaos system. Comparing performance between different algorithms is unnecessary and beyond the scope of this paper. The primary setting of all algorithms used in our experiments is reported in Table 3. Their meanings and additional parameters can be found in the original publications.

6.3. Parameter settings

Tests on 10$D$is carried out for the total of 55 problems, with the search range of $[-100, 100]^D$. The maximum number of function evaluations is 10 000$\times D$($MaxFEs$ = 100 000). Error value smaller than 10^{-8}will be taken as zero, as experimental settings requested in [61–63].

The Wilcoxon rank-sum test (WRT) was applied at the 5%significance level to evaluate whether the differences between the results are significant or not [69,70].

For PRGNs system: 10 algorithms were tested. For each algorithm, 55 functions were used. For each test function, 51 independent runs were performed. And for each independent run, 100 000 function evaluations were called (see Fig. 10).

For deterministic chaos system: 551 chaotic series were generated. For each chaotic series, 10 algorithms were tested. For each algorithm, 55 functions were used. For each test function, 51 independent runs were performed. And for each independent run, 100,000 function evaluations were called (see Fig. 11).

In 551 chaotic series: One chaotic series with full precision and 550 chaotic maps with limited precision. 550 chaotic maps were divided into 50 series and 11 levels of precisions. The difference between chaos series is the first number $x_0$created by the setting of *rng default* of Matlab (repeatable).

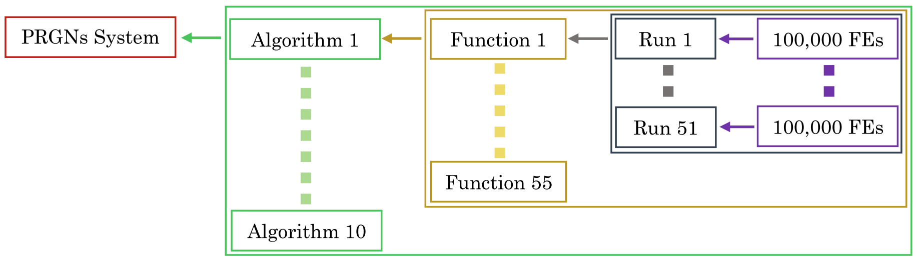

Fig. 10. The PRGNs system layout for the test.

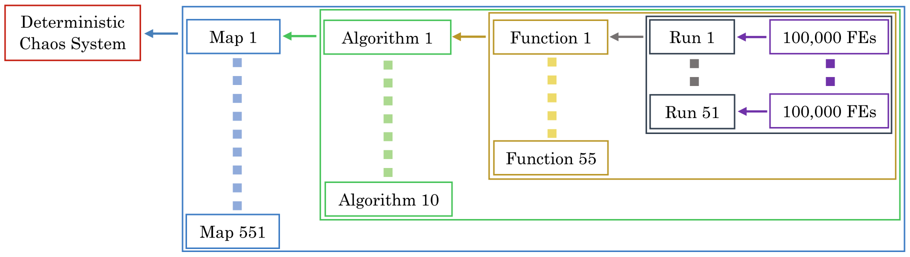

Fig. 11. The deterministic chaos system layout for the test.

6.4. Hardware infrastructure

All tests were run on the Salomon supercomputer (https://www.it4i.cz/en), under CentOS 7.x operating system, Matlab R2015b version (8.6.0.267246) 64-bit (glnxa64), using Parallel Computing Toolbox, within 20 compute nodes, 24 cores of two twelve-core Intel Haswell processors for each node (480 workers), and 2560 GB RAM in total.

The simulation consumed about 150,000 core hours for the computation, equivalent to around 17.36 continuously working years of the single CPU with 2.5 GHz.

7. Results

The results obtained from the description of the experiments in the previous section were visualized as done in the following figures. Since there was a large amount of data that needed to be displayed in some readable but also compressed way, we decided to display a graph for each algorithm, consisting of three colors. The *x*-axis shows the number of test functions used, the y-axis the accuracy with which the respective periodic series was generated. Remember that lower accuracy equals shorter chaotic series, which repeat earlier.

The green color represents those cases where the algorithm-driven by the classic pseudo-random number generator won. The yellow color represents those cases where, from the point of view of statistical evaluation, both algorithms had the same performance, and the blue color represents those cases in which the algorithm, which was powered by a chaotic number generator, won. Statistical results of the 50 chaotic series for each level of precision are visualized by the box plot located on each precision level. Thus, each figure represents a comparison of the selected algorithm with itself: one version is driven by a pseudo-random generator and the other version is driven by periodic series of different lengths produced via chaos generators by varying the already mentioned calculation accuracy. It is clear from the figure that the pseudo-random number generator clearly wins for low calculation accuracy. More specifically, if the accuracy of the chaotic series calculation is low, or the periodic series generated by chaos has a short period, then the algorithm that uses the pseudo-random generator gives better performance. However, it can be seen that from about precision 6, i.e. calculating to 6 decimal places, for many algorithms chaos begins to prevail and gives better results than the original algorithm with a pseudo-random number generator. In this case, a draw marked in yellow in the middle can be added to the good of chaos, because its presence did not impair the performance of the algorithm. However, in the evaluation better - worse we do not take this into account. In some algorithms, the benefit of chaos is quite significant, as it can be seen, for example, with the ACO algorithm or the cultural algorithm. The SOMA algorithm also shows strong signs of improvement over its classic canonical version. We note that this algorithm has many modern variants that have not been tested here. In the tables below, the exact numerical values and performance evaluations of the algorithm are reported, see Figs. 12,13 and Tables 4–13.

Evolutionary algorithms also work with pseudo-random integer numbers. During our experimentation, we had met some obstacles when chaotic series were used. Some algorithms employ the *randi* (uniformly distributed pseudo-random integers) to randomly select individuals in populations or subpopulations (a small number of individuals). For these algorithms, replacing *randi* with DCHS tends to work inefficiently (for example DE, see Fig. 13).

The reason is visualized in Fig. 14. The logistic equation return (for $A = 4$) values between 0–1, Fig. 14, on the left. For population members selection, we need integers in the interval [1,*N*]. So firstly, we just expanded the series into the interval [1,*N*] like [1,6] (on the right) which, for larger populations, shows itself as inefficient. We have found that an acceptable correction consists in expanding the chaotic series into intervals where N $\gg$population size, like [1], proven effectiveness in Table 4. Then integer needed for the individual selection from the population is given by the formula

$$Selected\_Individual = N\, modulo\, population\_size. \tag{3}$$

This has been verified as the efficient source of *chaotic integers*.

Let shortly summarize findings of chaotically generated integers:

- If the algorithm only uses *rand*: replace *rand* by deterministic chaos generator $\Rightarrow$better performance (than original)
- If the algorithm uses many types of random (for example: *rand*, *unifrnd*, *randn*):
  - just replace one type $\Rightarrow$ worse performance
  - replace all types $\Rightarrow$ better performance
- Replace integers - with small range [lower upper] $\Rightarrow$no change or worse performance (because it is no longer a chaotic map and repeat itself quickly).
- Replace integers – with huge range [lower upper] and use formula (3) $\Rightarrow$no change or better performance.

When evaluating the findings several interesting points arise. For example, the consideration of how this approach has changed or hasn't changed the complexity of the calculation. In our experiments, the canonical versions of the considered algorithms were used. Their structure was not affected in any way. The only thing that happened was that the numbers obtained from the random number generator were replaced by numbers obtained from the chaos generator, the course of which was reduced to periodicity with the accuracy of the calculation. So, the unique difference was how these numbers were calculated. Given that a more complex randomness generator was used on the one hand and a rather trivial mathemat-

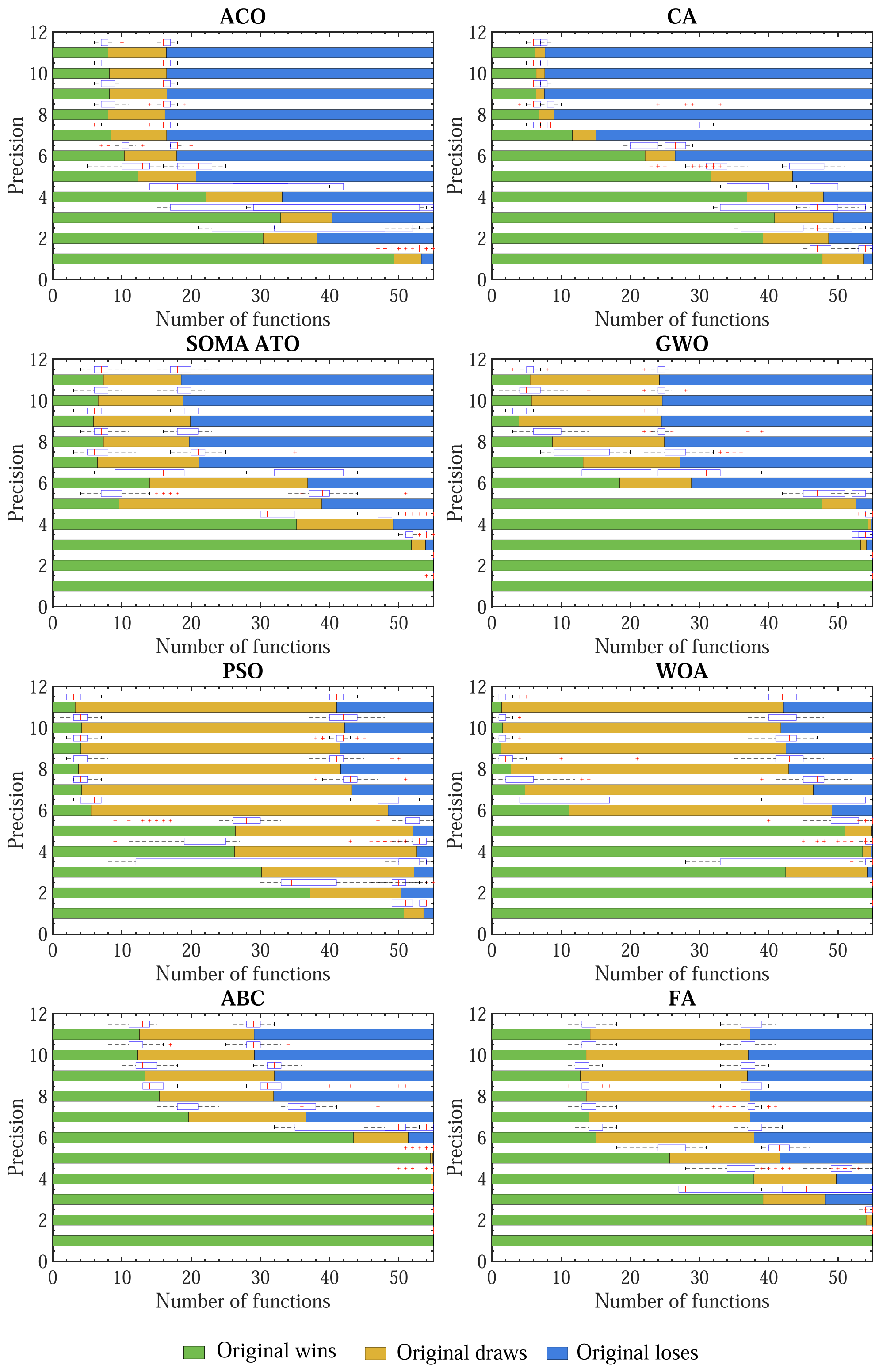

Fig. 12. The graphically visualized results on the IEEE CEC 2015, 2017, and 2020.

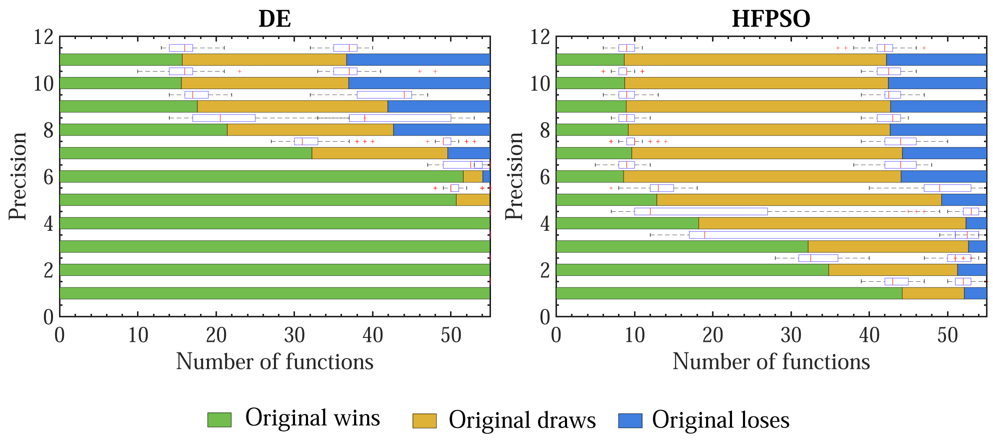

Fig. 13. The graphically visualized results on the IEEE CEC 2015, 2017, and 2020.

ical formula for generating chaos on the other, it is clear that the complexity of the algorithm certainly did not increase, as the calculation of the chaotic series itself is very trivial.

Regarding the advantages and disadvantages of the proposed approach, the following can be stated. Random number generators, if they are to meet modern requirements, are usually more complex algorithms that represent a mathematical relationship to generate pseudo-random numbers. On the other hand, if we look at discrete chaotic systems [71], it can be stated that their mathematical formulas are usually very trivial and they usually do not contain any complex mathematical operations or functions. Thus, it can be expected that the complexity of the algorithm does not increase with the use of chaotic generators, as discussed in the previous paragraph. It does not apply absolutely, but concerning the particular chaotic generator used. In the case of the logical equation used, it is clear that this is not a complex mathematical relationship that would burden the process of the new numerical simulation itself, which could be understood as an advantage of the proposed approach. The disadvantage is that deterministic chaos, as such, is a scientific field originating from physics and, despite its existence in the scientific world for many decades, a minor part of the computer science researchers community know it. This may limit its use and understanding in numerical experiments.

The achieved results will find wide applicability in many optimization problems. One of them, where our research is employed, lies in Quality of Service (QoS) routing in quantum key distribution (QKD) complex networks. Evolutionary algorithms help to solve the multi-constraints QoS routing problems.

The results presented and obtained here can be easily repeated by running our code, which can be found on the GitHub server[2]. We present only numerical observations and results, but it is clear that it is necessary to clarify the theoretical background. Why does this interesting effect occur when periodic series increase the performance of the evolutionary algorithm? We would like to keep this an open topic for further research.

8. Open questions and future research

According to the results presented in the results section, it can be stated that N periodic series clearly improved the performance of the algorithms, as demonstrated by statistical testing. Thus, it can be said that purely random processes are most likely unnecessary within these algorithms which have been demonstrated numerically. This finding is, after all, in line with the fact, namely that random processes are an unwelcome input that usually acts as disturbances in dynamic control systems, thus very rarely being beneficial. In other words, a dynamic that is subject to a process and has feedback should not contain randomness unless it is directly the goal of control. In control theory itself, random processes are understood usually as noise, and everything is done to erase/eliminate this component of behavior from the behavior of dynamic systems. Indeed, deliberate injections of (small) noise are normally used to trigger otherwise hidden dynamics, thus revealing the true nature of the system, or to prevent the control law from pathological failures induced by variables that stay at constant values for prolonged periods. This is typical of control techniques that need an accurate identification of the model (persistent excitability), and of systems that are prevented from operating when one or more variables are inside certain intervals (dead zone systems). However, when the classic control is in process (autopilot, reactor control, traffic control, etc) then the noise is not an acceptable signal. Thus, there is a discrepancy between what the evolutionary community still understands about the need for pseudo-random processes in evolutionary algorithms themselves and how randomness is understood in the control of dynamic systems (we recall that both evolutionary and swarm algorithms can be understood as discrete dynamic systems).

[2] https://github.com/Zelinkai/chaosimpact.git

Table 4
Comparison of DE algorithms (original vs chaos version with different PopSize) on the CEC 2017 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2017	PopSize = 50		PopSize = 200		PopSize = 500	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	1.24e + 03(1.63e + 03)	1.37e + 03(1.69e + 03) ≈	1.81e + 03(1.80e + 03)	1.54e + 03(1.69e + 03) ≈	6.44e + 03(5.94e + 03)	2.82e + 03(2.94e + 03)+
F_2	0.00e + 00(0.00e + 00)	0.00e + 00(0.00e + 00) ≈	0.00e + 00(0.00e + 00)	0.00e + 00(0.00e + 00) ≈	1.61e + 02(4.19e + 02)	4.54e + 01(6.92e + 01)+
F_3	0.00e + 00(0.00e + 00)	0.00e + 00(0.00e + 00) ≈	1.09e + 02(5.82e + 01)	2.98e + 01(2.70e + 01)+	1.51e + 03(4.54e + 02)	1.12e + 03(3.70e + 02)+
F_4	1.20e − 01(5.66e − 02)	2.19e − 01(3.45e − 01)−	2.14e + 00(5.69e − 01)	2.03e + 00(6.91e − 01) ≈	3.29e + 00(8.51e − 01)	3.51e + 00(6.54e − 01) ≈
F_5	1.00e + 01(4.67e + 00)	1.01e + 01(3.96e + 00) ≈	1.85e + 01(2.98e + 00)	6.34e + 00(2.88e + 00)+	2.14e + 01(3.99e + 00)	1.23e + 01(3.66e + 00)+
F_6	1.17e − 06(6.95e − 06)	8.46e − 05(4.03e − 04)−	0.00e + 00(0.00e + 00)	0.00e + 00(0.00e + 00) ≈	1.18e − 04(4.20e − 05)	1.24e − 05(4.68e − 06)+
F_7	2.52e + 01(3.27e + 00)	2.01e + 01(4.29e + 00)+	3.05e + 01(4.42e + 00)	1.55e + 01(3.35e + 00)+	3.46e + 01(4.28e + 00)	2.88e + 01(4.67e + 00)+
F_8	1.02e + 01(5.50e + 00)	9.13e + 00(3.09e + 00) ≈	1.90e + 01(3.58e + 00)	6.24e + 00(2.31e + 00)+	2.34e + 01(3.54e + 00)	1.43e + 01(4.58e + 00)+
F_9	0.00e + 00(0.00e + 00)	5.52e − 02(1.48e − 01)−	0.00e + 00(0.00e + 00)	0.00e + 00(0.00e + 00) ≈	1.57e − 07(1.04e − 07)	5.21e − 09(1.10e − 08)+
F_{10}	4.88e + 02(2.74e + 02)	4.10e + 02(2.15e + 02) ≈	8.53e + 02(1.74e + 02)	1.83e + 02(1.14e + 02)+	1.02e + 03(1.42e + 02)	6.14e + 02(2.55e + 02)+
F_{11}	1.17e + 00(1.27e + 00)	2.78e + 00(2.16e + 00)−	1.31e + 00(1.36e + 00)	1.85e + 00(1.62e + 00) ≈	5.78e + 00(1.15e + 00)	3.29e + 00(1.82e + 00)+
F_{12}	1.37e + 04(1.33e + 04)	1.37e + 04(1.41e + 04) ≈	2.41e + 04(1.84e + 04)	1.69e + 04(1.40e + 04)+	1.50e + 05(1.11e + 05)	6.02e + 04(6.56e + 04)+
F_{13}	8.38e + 02(8.80e + 02)	1.85e + 03(4.99e + 03) ≈	4.34e + 02(4.71e + 02)	1.06e + 03(9.15e + 02)−	3.15e + 02(2.46e + 02)	5.49e + 02(5.19e + 02) ≈
F_{14}	1.15e + 01(1.81e + 01)	1.61e + 02(4.77e + 02)−	1.20e + 01(4.19e + 00)	5.45e + 00(7.06e + 00)+	2.22e + 01(3.95e + 00)	1.08e + 01(6.25e + 00)+
F_{15}	1.03e + 00(1.20e + 00)	3.54e + 02(2.24e + 03)−	3.77e + 00(2.18e + 00)	4.43e + 00(8.66e + 00)−	9.46e + 00(2.70e + 00)	5.09e + 00(3.43e + 00)+
F_{16}	7.90e + 00(2.92e + 01)	7.71e + 01(1.03e + 02)−	7.75e − 01(1.56e + 00)	8.05e + 00(2.51e + 01) ≈	1.43e + 00(7.48e − 01)	2.13e + 00(3.83e + 00)−
F_{17}	4.56e + 00(9.72e + 00)	1.67e + 01(2.41e + 01)−	1.71e + 00(3.96e + 00)	3.97e + 00(7.07e + 00)−	1.29e + 01(4.08e + 00)	2.96e + 00(5.11e + 00)+
F_{18}	1.14e + 02(1.93e + 02)	2.06e + 03(4.37e + 03) ≈	1.05e + 02(1.33e + 02)	7.95e + 01(1.72e + 02)+	8.76e + 02(1.47e + 03)	3.34e + 02(6.32e + 02)+
F_{19}	4.23e − 01(5.04e − 01)	6.34e + 02(1.89e + 03)−	1.23e + 00(1.36e + 00)	1.01e + 00(2.99e + 00)+	4.89e + 00(9.85e − 01)	2.42e + 00(2.20e + 00)+
F_{20}	2.00e + 00(4.30e + 00)	7.86e + 00(1.80e + 01)−	4.28e − 01(4.45e − 01)	9.32e − 01(1.92e + 00)−	2.89e − 01(2.83e − 01)	7.10e − 01(6.61e − 01) ≈
F_{21}	1.91e + 02(4.84e + 01)	1.65e + 02(5.68e + 01)+	1.40e + 02(5.72e + 01)	1.56e + 02(5.50e + 01) ≈	1.33e + 02(5.33e + 01)	1.22e + 02(4.33e + 01)+
F_{22}	9.36e + 01(2.55e + 01)	1.01e + 02(7.77e − 01)−	8.69e + 01(3.67e + 01)	9.42e + 01(2.47e + 01)−	8.98e + 01(3.49e + 01)	8.88e + 01(3.24e + 01)+
F_{23}	3.07e + 02(3.61e + 00)	3.11e + 02(5.75e + 00)−	3.17e + 02(4.13e + 00)	3.08e + 02(2.97e + 00)+	3.21e + 02(4.43e + 00)	3.12e + 02(4.43e + 00)+
F_{24}	3.13e + 02(7.88e + 01)	3.22e + 02(6.57e + 01) ≈	3.16e + 02(8.72e + 01)	2.95e + 02(9.73e + 01)+	2.74e + 02(1.20e + 02)	2.60e + 02(1.19e + 02)+
F_{25}	4.20e + 02(2.32e + 01)	4.28e + 02(2.30e + 01)−	4.20e + 02(2.32e + 01)	4.19e + 02(2.33e + 01) ≈	4.21e + 02(2.33e + 01)	4.18e + 02(2.34e + 01) ≈
F_{26}	3.00e + 02(1.70e + 01)	3.10e + 02(2.20e + 01)−	2.94e + 02(4.20e + 01)	2.96e + 02(4.33e + 01) ≈	3.00e + 02(6.42e − 06)	3.00e + 02(7.77e − 06)+
F_{27}	3.93e + 02(2.70e + 00)	3.94e + 02(3.16e + 00)−	3.91e + 02(2.90e + 00)	3.92e + 02(2.75e + 00) ≈	3.91e + 02(2.38e + 00)	3.91e + 02(2.88e + 00) ≈
F_{28}	4.07e + 02(1.39e + 02)	4.14e + 02(1.53e + 02) ≈	3.21e + 02(7.53e + 01)	3.26e + 02(8.20e + 01) ≈	3.01e + 02(1.11e + 00)	3.09e + 02(2.91e + 01) ≈
F_{29}	2.64e + 02(8.08e + 00)	2.57e + 02(2.10e + 01)+	2.74e + 02(1.01e + 01)	2.60e + 02(1.28e + 01)+	2.84e + 02(1.25e + 01)	2.78e + 02(1.52e + 01)+
F_{30}	1.34e + 05(2.98e + 05)	1.49e + 05(3.18e + 05)−	3.46e + 04(1.22e + 05)	6.86e + 04(2.21e + 05)−	5.49e + 04(4.26e + 04)	4.26e + 04(1.56e + 05)+
Chaos wins	+	3	+	12	+	23
Original wins	−	16	−	6	−	1
Similar	≈	11	≈	12	≈	6

Table 5
Comparison of algorithms (SOMA ATO, HFPSO, and ABC original vs chaos version) on the CEC 2017 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2017	SOMA ATO		HFPSO		ABC	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	1.66e+03(2.12e+03)	1.14e+01(1.61e+01)+	2.57e+03(2.94e+03)	3.62e+03(4.10e+03) ≈	5.91e+02(6.18e+02)	1.29e+04(2.30e+04)−
F_2	1.43e+00(7.09e+00)	0.00e+00(0.00e+00)+	7.37e+00(3.46e+01)	6.28e+02(4.44e+03) ≈	2.49e+06(3.09e+06)	1.78e+06(7.67e+06)+
F_3	1.67e+03(9.72e+02)	7.57e−02(1.21e−01)+	0.00e+00(0.00e+00)	0.00e+00(0.00e+00) ≈	1.39e+04(3.89e+03)	1.21e+03(2.90e+03)+
F_4	4.02e+00(1.73e+00)	3.96e+00(1.28e+00) ≈	1.31e+00(7.76e+00)	1.47e+00(5.45e−01)−	3.89e+00(1.40e−01)	7.04e+00(1.38e+01)−
F_5	7.15e+00(1.60e+00)	4.55e+00(1.68e+00)+	1.64e+01(6.01e+00)	1.09e+01(5.18e+00)+	2.73e+01(3.74e+00)	1.92e+01(8.85e+00)+
F_6	2.13e−05(6.60e−06)	2.54e−07(1.62e−06)+	9.78e−02(4.08e−01)	1.07e−02(7.61e−02) ≈	3.12e−06(6.35e−06)	5.76e−02(2.13e−01)−
F_7	1.94e+01(2.39e+00)	1.59e+01(3.56e+00)+	1.79e+01(3.87e+00)	1.90e+01(3.18e+00) ≈	3.82e+01(4.43e+00)	2.78e+01(8.51e+00)+
F_8	7.54e+00(2.01e+00)	4.19e+00(1.49e+00)+	1.27e+01(6.14e+00)	8.74e+00(2.87e+00)+	2.64e+01(3.85e+00)	1.77e+01(7.48e+00)+
F_9	7.37e−07(1.10e−06)	0.00e+00(0.00e+00)+	0.00e+00(0.00e+00)	0.00e+00(0.00e+00) ≈	0.00e+00(0.00e+00)	2.35e−01(6.46e−01)−
F_{10}	3.23e+02(1.00e+02)	1.48e+02(1.17e+02)+	5.42e+02(2.10e+02)	4.07e+02(2.36e+02)+	1.45e+03(1.18e+02)	8.40e+02(3.78e+02)+
F_{11}	3.29e+00(1.26e+00)	2.56e+00(1.81e+00)+	9.48e+00(9.19e+00)	5.91e+00(3.78e+00)+	6.87e+00(1.11e+00)	9.19e+00(5.24e+00) ≈
F_{12}	2.88e+04(4.39e+04)	1.35e+04(1.33e+04)+	1.56e+04(1.36e+04)	1.43e+04(1.28e+04) ≈	2.52e+06(1.29e+06)	2.58e+04(3.06e+04)+
F_{13}	2.41e+02(3.52e+02)	8.37e+00(3.51e+00)+	5.84e+03(6.44e+03)	5.94e+03(6.08e+03) ≈	1.12e+04(3.03e+03)	1.07e+04(1.62e+04)+
F_{14}	1.63e+00(9.68e−01)	1.61e+00(1.18e+00) ≈	6.43e+01(3.50e+01)	5.14e+01(2.14e+01) ≈	7.21e+02(4.77e+02)	4.58e+01(2.66e+01)+
F_{15}	1.14e+00(6.25e−01)	7.76e−01(7.50e−01)+	4.87e+01(5.87e+01)	3.49e+01(3.12e+01) ≈	3.62e+03(1.90e+03)	6.33e+02(9.63e+02)+
F_{16}	2.02e+00(3.13e+00)	2.08e+00(3.07e+00) ≈	2.44e+02(1.28e+02)	2.32e+02(1.44e+02) ≈	3.97e+01(1.83e+01)	7.09e+01(6.97e+01) ≈
F_{17}	2.01e+00(3.28e+00)	1.43e+00(2.51e+00)+	5.24e+01(3.43e+01)	4.24e+01(3.27e+01) ≈	4.61e+01(6.56e+00)	4.56e+01(2.00e+01)+
F_{18}	4.34e+00(3.03e+00)	5.46e+00(1.04e+01)−	1.11e+04(1.06e+04)	1.06e+04(1.04e+04) ≈	6.69e+04(4.69e+04)	6.70e+03(1.54e+04)+
F_{19}	2.97e−01(3.63e−01)	1.13e+01(1.55e+01)−	1.16e+02(2.53e+02)	3.55e+02(1.82e+03)−	9.93e+02(5.90e+02)	6.61e+02(1.22e+03)+
F_{20}	9.94e−02(2.20e−01)	2.09e−01(3.85e−01)−	7.02e+01(5.92e+01)	8.26e+01(6.57e+01) ≈	2.41e+01(2.07e+00)	2.60e+01(7.60e+00) ≈
F_{21}	1.07e+02(8.82e+00)	9.90e+01(2.50e+01)+	1.97e+02(4.53e+01)	1.85e+02(5.00e+01)+	1.65e+02(2.11e+01)	2.07e+02(4.15e+01)−
F_{22}	6.56e+01(3.49e+01)	7.24e+01(4.16e+01) ≈	1.02e+02(1.87e+01)	1.15e+02(9.84e+01) ≈	1.03e+02(1.45e+00)	2.03e+02(3.10e+02) ≈
F_{23}	3.11e+02(2.92e+00)	3.02e+02(4.30e+01)+	3.26e+02(1.31e+01)	3.18e+02(7.45e+00)+	3.30e+02(4.13e+00)	3.25e+02(1.09e+01)+
F_{24}	1.20e+02(4.35e+01)	1.26e+02(7.37e+01)−	3.03e+02(1.02e+02)	3.21e+02(8.19e+01) ≈	3.53e+02(2.08e+01)	3.57e+02(9.11e+00) ≈
F_{25}	4.01e+02(8.14e+00)	4.19e+02(2.30e+01)−	4.22e+02(5.28e+01)	4.05e+02(6.63e+01) ≈	4.35e+02(1.18e+01)	4.28e+02(2.50e+01) ≈
F_{26}	2.28e+02(1.17e+02)	2.75e+02(8.49e+01)−	4.60e+02(3.60e+02)	3.22e+02(2.39e+02)+	4.89e+02(8.49e+01)	5.40e+02(1.51e+02) ≈
F_{27}	3.92e+02(2.37e+00)	3.93e+02(2.77e+00) ≈	4.10e+02(2.63e+01)	4.02e+02(2.50e+01)+	4.57e+02(3.24e+01)	4.77e+02(3.40e+01)−
F_{28}	2.66e+02(9.05e+01)	2.62e+02(9.80e+01)+	4.80e+02(1.43e+02)	4.64e+02(1.44e+02) ≈	4.92e+02(6.59e+00)	4.87e+02(1.27e+01) ≈
F_{29}	2.63e+02(8.87e+00)	2.51e+02(7.32e+00)+	3.02e+02(5.65e+01)	2.84e+02(3.54e+01) ≈	3.42e+02(2.20e+01)	3.32e+02(5.85e+01) ≈
F_{30}	3.57e+04(4.78e+04)	3.83e+03(4.29e+03)+	5.06e+05(8.52e+05)	2.81e+05(5.13e+05) ≈	7.15e+02(3.50e+02)	1.95e+03(2.50e+03)−
Chaos wins	+	19	+	8	+	14
Original wins	−	6	−	2	−	7
Similar	≈	5	≈	20	≈	9

Table 6
Comparison of algorithms (GWO, WOA, and ACOR original vs chaos version) on the CEC 2017 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2017	GWO		WOA		ACOR	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	3.08e+08(3.98e+08)	9.42e+07(1.68e+08)+	3.93e+05(8.14e+05)	1.88e+05(2.73e+05) ≈	1.90e+03(2.31e+03)	7.85e+02(1.18e+03)+
F_2	1.52e+09(3.01e+09)	3.75e+07(2.31e+08)+	2.10e+04(3.19e+04)	1.37e+04(3.25e+04) ≈	7.96e+05(5.03e+06)	0.00e+00(0.00e+00)+
F_3	1.78e+04(9.02e+03)	1.14e+03(1.87e+03)+	5.14e+02(7.52e+02)	2.67e+02(3.37e+02) ≈	8.02e+02(1.34e+03)	0.00e+00(0.00e+00)+
F_4	3.67e+01(2.39e+01)	1.47e+01(1.79e+01)+	1.78e+01(2.80e+01)	2.08e+01(3.35e+01) ≈	2.16e+00(1.38e−01)	2.88e+00(3.95e−01)−
F_5	3.63e+01(1.45e+01)	1.65e+01(8.30e+00)+	5.34e+01(1.78e+01)	4.52e+01(1.67e+01)+	2.86e+01(4.64e+00)	2.01e+01(6.40e+00)+
F_6	7.76e+00(7.55e+00)	1.14e+00(1.25e+00)+	3.05e+01(1.30e+01)	2.93e+01(1.28e+01) ≈	0.00e+00(0.00e+00)	2.22e−07(1.59e−06) ≈
F_7	5.68e+01(1.62e+01)	3.31e+01(1.02e+01)+	7.78e+01(2.21e+01)	8.02e+01(2.47e+01) ≈	3.81e+01(4.78e+00)	3.10e+01(3.31e+00)+
F_8	3.14e+01(1.56e+01)	1.40e+01(6.49e+00)+	3.98e+01(1.24e+01)	3.93e+01(1.60e+01) ≈	2.88e+01(4.62e+00)	2.02e+01(4.74e+00)+
F_9	1.18e+02(2.36e+02)	1.62e+01(3.90e+01)+	4.13e+02(2.81e+02)	4.71e+02(3.46e+02) ≈	0.00e+00(0.00e+00)	8.91e−03(6.36e−02) ≈
F_{10}	1.46e+03(5.83e+02)	4.97e+02(2.82e+02)+	1.02e+03(3.31e+02)	9.35e+02(3.55e+02) ≈	1.70e+03(1.18e+02)	1.29e+03(1.70e+02)+
F_{11}	3.90e+02(1.05e+03)	3.08e+01(2.11e+01)+	1.09e+02(9.81e+01)	9.69e+01(7.19e+01) ≈	6.55e+00(1.51e+00)	7.61e−01(7.11e−01)+
F_{12}	3.05e+06(3.51e+06)	4.21e+05(6.65e+05)+	3.83e+06(5.03e+06)	3.88e+06(5.00e+06) ≈	2.79e+04(8.38e+04)	7.38e+03(3.64e+03)+
F_{13}	2.01e+04(1.42e+04)	1.14e+04(8.40e+03)+	1.43e+04(1.26e+04)	1.44e+04(1.17e+04) ≈	8.92e+03(7.91e+03)	1.10e+04(5.52e+03)−
F_{14}	6.26e+03(6.96e+03)	5.27e+02(1.11e+03)+	4.55e+02(9.68e+02)	2.25e+02(4.04e+02)+	8.92e+02(8.66e+02)	3.26e+03(2.67e+03)−
F_{15}	1.91e+04(1.74e+04)	1.30e+03(1.43e+03)+	2.39e+03(2.08e+03)	2.59e+03(2.86e+03) ≈	6.19e+03(3.33e+03)	1.54e+03(1.78e+03)+
F_{16}	2.47e+02(1.39e+02)	1.41e+02(1.29e+02)+	2.65e+02(1.36e+02)	2.18e+02(1.28e+02) ≈	3.93e+01(2.33e+01)	3.57e+00(8.41e+00)+
F_{17}	1.34e+02(6.42e+01)	4.84e+01(1.58e+01)+	8.46e+01(3.69e+01)	9.54e+01(4.75e+01) ≈	4.24e+01(1.07e+01)	3.89e+01(1.14e+01)+
F_{18}	7.03e+04(4.89e+04)	2.97e+04(1.42e+04)+	1.69e+04(1.21e+04)	1.27e+04(1.23e+04)+	1.03e+05(7.94e+04)	6.94e+03(6.31e+03)+
F_{19}	6.53e+04(1.33e+05)	3.01e+03(4.75e+03)+	1.75e+04(3.92e+04)	1.60e+04(2.31e+04) ≈	2.60e+03(2.64e+03)	4.17e+03(3.27e+03)−
F_{20}	1.87e+02(9.43e+01)	6.59e+01(4.30e+01)+	1.55e+02(7.46e+01)	1.43e+02(6.95e+01) ≈	2.04e+01(2.11e−01)	2.29e+01(5.39e+00) ≈
F_{21}	2.35e+02(2.72e+01)	2.08e+02(3.17e+01)+	2.24e+02(5.63e+01)	2.06e+02(6.51e+01) ≈	2.29e+02(8.25e+00)	2.23e+02(5.58e+00)+
F_{22}	1.53e+02(2.13e+02)	1.17e+02(2.54e+01)+	1.31e+02(1.24e+02)	1.33e+02(1.34e+02) ≈	2.06e+02(3.70e+02)	1.00e+02(2.52e−01)+
F_{23}	3.49e+02(1.81e+01)	3.20e+02(9.43e+00)+	3.49e+02(2.00e+01)	3.38e+02(1.91e+01)+	3.33e+02(4.79e+00)	3.17e+02(8.42e+00)+
F_{24}	3.71e+02(3.72e+01)	3.45e+02(1.27e+01)+	3.54e+02(8.24e+01)	3.64e+02(4.05e+01) ≈	3.60e+02(4.26e+00)	3.50e+02(6.18e+00)+
F_{25}	4.51e+02(1.81e+01)	4.29e+02(1.46e+01)+	4.36e+02(5.26e+01)	4.41e+02(2.96e+01) ≈	4.42e+02(1.31e+01)	4.46e+02(7.01e+00)−
F_{26}	7.88e+02(4.17e+02)	4.81e+02(3.43e+02)+	8.87e+02(5.71e+02)	6.88e+02(5.28e+02)+	6.92e+02(6.70e+01)	5.03e+02(1.46e+02)+
F_{27}	4.20e+02(2.81e+01)	3.94e+02(1.06e+01)+	4.34e+02(3.74e+01)	4.10e+02(2.32e+01)+	4.81e+02(2.15e+01)	4.09e+02(4.83e+01)+
F_{28}	6.40e+02(1.23e+02)	6.00e+02(4.58e+01)+	6.36e+02(1.68e+02)	5.93e+02(6.37e+01)+	4.96e+02(8.08e+00)	4.73e+02(3.75e+00)+
F_{29}	3.75e+02(6.51e+01)	2.88e+02(3.91e+01)+	4.27e+02(7.80e+01)	3.97e+02(8.64e+01) ≈	3.45e+02(4.14e+01)	2.73e+02(9.72e+00)+
F_{30}	4.07e+06(5.31e+06)	7.01e+05(7.75e+05)+	5.34e+05(9.98e+05)	2.54e+05(4.20e+05) ≈	1.96e+03(1.57e+03)	2.47e+02(2.29e+01)+
Chaos wins	+	30	+	7	+	22
Original wins	−	0	−	0	−	5
Similar	≈	0	≈	23	≈	3

Table 7
Comparison of algorithms (FA, PSO, and CA original vs chaos version) on the CEC 2017 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2017	FA		PSO		CA	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	3.85e+03(4.27e+03)	4.19e+03(3.63e+03) ≈	1.79e+03(1.66e+03)	2.41e+03(2.59e+03) ≈	4.36e+03(6.32e+03)	1.45e+07(2.48e+07)−
F_2	3.92e−02(1.96e−01)	0.00e+00(0.00e+00) ≈	0.00e+00(0.00e+00)	0.00e+00(0.00e+00) ≈	9.71e+83(8.71e+68)	4.13e+63(2.95e+64)+
F_3	1.24e−05(3.27e−06)	9.16e−04(2.57e−04)−	0.00e+00(0.00e+00)	0.00e+00(0.00e+00) ≈	4.85e+04(2.38e+04)	3.29e+04(1.03e+04)+
F_4	2.78e+00(1.12e+00)	1.92e+00(3.35e−01)+	2.64e+00(9.39e+00)	2.84e+00(7.52e−01)−	5.93e+00(2.06e−01)	8.53e+00(8.19e−01)−
F_5	1.74e+01(8.18e+00)	8.99e+00(3.90e+00)+	1.69e+01(7.95e+00)	1.49e+01(7.50e+00) ≈	4.23e+01(5.02e+00)	1.27e+01(6.22e+00)+
F_6	1.58e−03(2.38e−04)	1.40e−02(1.89e−03)−	3.83e−01(8.88e−01)	1.83e−01(4.12e−01)+	6.86e+01(8.81e+00)	1.74e−06(7.56e−06)+
F_7	1.73e+01(3.01e+00)	1.71e+01(2.96e+00) ≈	1.91e+01(4.29e+00)	1.72e+01(4.05e+00)+	5.21e+01(5.23e+00)	2.41e+01(6.61e+00)+
F_8	1.32e+01(5.13e+00)	7.24e+00(3.01e+00)+	1.29e+01(5.18e+00)	1.03e+01(3.77e+00)+	4.32e+01(6.31e+00)	1.16e+01(6.01e+00)+
F_9	2.67e−06(7.27e−07)	2.07e−04(5.38e−05)−	8.91e−03(6.36e−02)	0.00e+00(0.00e+00) ≈	9.31e+00(1.93e+01)	1.92e−02(4.83e−02)+
F_{10}	6.07e+02(3.30e+02)	2.36e+02(1.67e+02)+	6.55e+02(2.53e+02)	5.48e+02(2.30e+02)+	2.09e+03(1.93e+02)	6.73e+02(3.63e+02)+
F_{11}	5.95e+00(3.28e+00)	4.61e+00(2.77e+00)+	1.76e+01(1.09e+01)	1.09e+01(6.78e+00)+	3.33e+03(2.33e+03)	7.96e+02(1.09e+03)+
F_{12}	1.34e+04(1.18e+04)	1.65e+04(1.62e+04) ≈	9.25e+03(7.31e+03)	1.40e+04(1.01e+04)−	3.23e+08(1.67e+08)	5.72e+05(9.68e+05)+
F_{13}	4.20e+03(5.32e+03)	4.06e+03(5.03e+03) ≈	6.56e+03(5.15e+03)	5.55e+03(4.62e+03) ≈	1.28e+06(1.38e+06)	1.59e+04(5.38e+03)+
F_{14}	4.95e+01(8.45e+01)	2.52e+01(1.15e+01)+	5.48e+01(2.17e+01)	5.55e+01(2.27e+01) ≈	8.22e+03(8.10e+03)	2.43e+03(1.50e+03)+
F_{15}	8.59e+01(3.31e+02)	2.76e+01(2.37e+01) ≈	6.18e+01(4.84e+01)	4.06e+01(2.43e+01)+	5.27e+04(4.49e+04)	3.75e+03(3.19e+03)+
F_{16}	7.22e+01(7.74e+01)	1.37e+01(1.81e+01)+	2.09e+02(1.30e+02)	1.69e+02(1.40e+02)+	2.14e+02(5.76e+01)	1.36e+01(1.72e+01)+
F_{17}	2.71e+01(1.88e+01)	2.61e+01(1.15e+01) ≈	5.27e+01(3.07e+01)	4.73e+01(3.25e+01) ≈	9.29e+01(2.01e+01)	4.38e+01(1.47e+01)+
F_{18}	7.66e+03(7.19e+03)	1.07e+04(9.04e+03) ≈	4.18e+03(6.40e+03)	4.01e+03(6.06e+03) ≈	8.54e+06(1.06e+07)	1.33e+04(1.40e+04)+
F_{19}	2.45e+02(7.24e+02)	1.65e+01(1.75e+01)+	2.13e+02(5.35e+02)	1.01e+02(1.90e+02) ≈	5.76e+04(9.30e+04)	4.91e+03(3.56e+03)+
F_{20}	1.62e+01(1.57e+01)	1.11e+01(2.14e+01)+	6.79e+01(5.54e+01)	6.28e+01(5.62e+01) ≈	1.88e+02(3.46e+01)	5.89e+01(1.27e+01)+
F_{21}	1.72e+02(5.55e+01)	1.65e+02(5.85e+01) ≈	1.77e+02(5.51e+01)	1.70e+02(5.65e+01) ≈	2.38e+02(1.99e+01)	2.09e+02(1.79e+01)+
F_{22}	9.79e+01(1.59e+01)	9.00e+01(3.08e+01) ≈	1.02e+02(8.75e−01)	1.12e+02(9.49e+01) ≈	1.41e+03(4.13e+02)	1.14e+02(1.02e+02)+
F_{23}	3.18e+02(6.82e+00)	3.10e+02(3.26e+00)+	3.20e+02(8.10e+00)	3.17e+02(9.27e+00)+	3.64e+02(9.14e+00)	3.16e+02(5.58e+00)+
F_{24}	2.98e+02(9.91e+01)	3.40e+02(3.78e+00)−	2.94e+02(9.88e+01)	2.84e+02(1.03e+02) ≈	3.79e+02(4.82e+00)	3.49e+02(7.08e+00)+
F_{25}	4.22e+02(2.32e+01)	4.27e+02(2.33e+01) ≈	4.21e+02(2.33e+01)	4.20e+02(2.36e+01) ≈	4.58e+02(4.97e+00)	4.46e+02(6.92e+00)+
F_{26}	3.30e+02(6.13e+01)	3.00e+02(3.83e−03) ≈	2.87e+02(3.95e+01)	3.26e+02(2.07e+02) ≈	9.11e+02(6.73e+01)	5.28e+02(1.04e+02)+
F_{27}	3.96e+02(3.87e+00)	3.90e+02(7.33e−01)+	4.07e+02(2.78e+01)	4.01e+02(1.88e+01) ≈	4.74e+02(2.32e+01)	4.45e+02(2.14e+01)+
F_{28}	4.14e+02(1.38e+02)	4.26e+02(1.52e+02)−	4.53e+02(1.47e+02)	4.71e+02(1.48e+02) ≈	4.98e+02(2.65e+00)	4.98e+02(1.66e+00) ≈
F_{29}	2.77e+02(3.22e+01)	2.42e+02(8.48e+00)+	3.05e+02(3.84e+01)	2.87e+02(3.84e+01)+	6.81e+02(1.28e+02)	4.65e+02(1.51e+02)+
F_{30}	1.88e+05(3.60e+05)	2.67e+05(3.99e+05) ≈	8.43e+04(2.44e+05)	1.19e+05(3.19e+05) ≈	7.97e+05(8.59e+05)	2.95e+03(2.68e+03)+
Chaos wins	+	12	+	9	+	27
Original wins	−	5	−	2	−	2
Similar	≈	13	≈	19	≈	1

Table 8
Comparison of algorithms (SOMA ATO, HFPSO, and ABC original vs chaos version) on the CEC 2020 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2020	SOMA ATO		HFPSO		ABC	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	$1.31e+03(1.83e+03)$	$2.76e+01(4.58e+01)$+	$1.01e+03(1.12e+03)$	$2.49e+03(3.51e+03)\approx$	$9.33e+02(1.33e+03)$	$1.06e+04(1.71e+04)-$
F_2	$1.77e+02(8.09e+01)$	$6.93e+01(6.87e+01)$+	$3.06e+02(2.68e+02)$	$2.18e+02(1.24e+02)\approx$	$1.37e+03(1.59e+02)$	$7.29e+02(2.96e+02)$+
F_3	$1.97e+01(2.53e+00)$	$1.67e+01(2.94e+00)$+	$1.46e+01(6.65e-01)$	$1.99e+01(4.60e+00)-$	$3.71e+01(3.68e+00)$	$2.76e+01(7.80e+00)$+
F_4	$1.50e+00(3.16e-01)$	$9.70e-01(3.27e-01)$+	$6.79e-01(1.35e-01)$	$1.56e+00(3.67e+00)-$	$1.83e+00(3.58e-01)$	$1.52e+00(6.22e-01)$+
F_5	$1.90e+04(2.40e+04)$	$5.87e+02(1.01e+03)$+	$3.39e+03(2.44e+03)$	$2.46e+03(2.64e+03)\approx$	$5.13e+04(2.65e+04)$	$1.20e+04(1.63e+04)$+
F_6	$4.14e+00(1.68e+01)$	$1.72e+00(2.98e+00)\approx$	$1.24e+02(1.60e+02)$	$1.65e+02(8.01e+01)-$	$7.50e+01(2.90e+01)$	$1.12e+02(8.06e+01)-$
F_7	$7.59e+02(1.36e+03)$	$5.48e+00(1.10e+01)$+	$2.22e+02(2.52e+02)$	$1.39e+02(1.48e+02)\approx$	$1.45e+04(7.35e+03)$	$6.00e+02(6.49e+02)$+
F_8	$7.69e+01(3.42e+01)$	$7.70e+01(3.73e+01)\approx$	$1.01e+02(1.53e-01)$	$1.23e+02(1.27e+02)-$	$1.03e+02(1.19e+00)$	$1.74e+02(1.80e+02)\approx$
F_9	$1.24e+02(3.74e+01)$	$1.31e+02(7.19e+01)-$	$3.54e+02(9.20e+00)$	$3.29e+02(6.83e+01)$+	$3.45e+02(2.85e+01)$	$3.58e+02(1.48e+01)\approx$
F_{10}	$4.01e+02(1.10e+01)$	$4.14e+02(2.19e+01)\approx$	$4.20e+02(2.33e+01)$	$4.26e+02(2.36e+01)-$	$4.30e+02(1.37e+01)$	$4.21e+02(3.94e+01)\approx$
Chaos wins	+	6	+	1	+	5
Original wins	–	1	–	5	–	2
Similar	$\approx$	3	$\approx$	4	$\approx$	3

Table 9
Comparison of algorithms (GWO, WOA, and ACOR original vs chaos version) on the CEC 2020 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2020	GWO		WOA		ACOR	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	2.12e+07(9.72e+07)	2.70e+07(9.83e+07)–	3.34e+05(7.26e+05)	2.14e+05(3.40e+05) ≈	8.27e+02(9.97e+02)	1.16e+03(1.91e+03) ≈
F_2	3.82e+02(2.63e+02)	4.48e+02(2.68e+02) ≈	9.65e+02(2.77e+02)	9.69e+02(2.56e+02) ≈	1.60e+03(1.86e+02)	1.26e+03(2.01e+02)+
F_3	2.83e+01(9.54e+00)	2.91e+01(8.09e+00) ≈	8.24e+01(2.49e+01)	7.28e+01(2.21e+01) ≈	4.09e+01(2.05e+00)	2.97e+01(3.78e+00)+
F_4	3.29e+00(1.07e+01)	1.76e+00(9.01e−01) ≈	6.23e+00(4.34e+00)	5.17e+00(2.95e+00) ≈	2.62e+00(2.10e−01)	1.74e+00(3.52e−01)+
F_5	4.50e+04(1.12e+05)	5.59e+03(4.72e+03) ≈	1.39e+05(2.82e+05)	8.35e+04(1.33e+05) ≈	2.27e+04(1.48e+04)	6.60e+04(5.87e+04)–
F_6	1.32e+02(9.58e+01)	1.38e+02(7.97e+01) ≈	2.00e+02(1.01e+02)	1.74e+02(8.80e+01) ≈	5.97e+01(3.59e+01)	7.33e+00(1.79e+01)+
F_7	5.49e+03(4.61e+03)	4.30e+03(3.85e+03) ≈	1.80e+04(1.39e+04)	1.70e+04(1.43e+04) ≈	3.58e+03(1.19e+03)	1.15e+04(8.56e+03)–
F_8	1.21e+02(5.99e+01)	1.26e+02(7.78e+01)–	1.74e+02(2.35e+02)	1.78e+02(2.65e+02) ≈	1.04e+02(6.33e−01)	1.00e+02(2.83e−01)+
F_9	3.45e+02(1.03e+01)	3.47e+02(1.19e+01) ≈	3.72e+02(4.53e+01)	3.66e+02(4.19e+01) ≈	3.64e+02(5.84e+00)	3.51e+02(5.62e+00)+
F_{10}	4.30e+02(1.83e+01)	4.35e+02(1.85e+01) ≈	4.26e+02(6.44e+01)	4.33e+02(6.40e+01) ≈	4.24e+02(2.42e+01)	4.44e+02(1.17e+01)–
Chaos wins	+	0	+	0	+	6
Original wins	–	2	–	0	–	3
Similar	≈	8	≈	10	≈	1

Table 10
Comparison of algorithms FA, PSO, and CA original vs chaos version) on the CEC 2020 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2020	FA		PSO		CA	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	$3.58e+03(3.91e+03)$	$6.80e+03(3.93e+03)-$	$2.49e+03(3.16e+03)$	$2.35e+03(2.87e+03)\approx$	$3.53e+03(2.46e+03)$	$4.40e+07(1.47e+08)-$
F_2	$5.44e+02(2.90e+02)$	$2.40e+02(1.45e+02)+$	$4.22e+02(2.61e+02)$	$3.94e+02(2.05e+02)\approx$	$2.12e+03(1.19e+02)$	$6.01e+02(2.67e+02)+$
F_3	$1.79e+01(4.43e+00)$	$1.66e+01(3.70e+00)+$	$1.97e+01(4.86e+00)$	$1.79e+01(3.53e+00)+$	$5.46e+01(1.96e+00)$	$2.35e+01(5.83e+00)+$
F_4	$8.36e-01(3.31e-01)$	$8.38e-01(2.58e-01)\approx$	$9.47e-01(3.22e-01)$	$9.37e-01(3.28e-01)\approx$	$3.15e+00(5.54e-01)$	$9.37e-01(5.22e-01)+$
F_5	$2.40e+03(2.76e+03)$	$4.31e+03(4.26e+03)\approx$	$2.30e+03(1.98e+03)$	$2.26e+03(2.14e+03)\approx$	$2.91e+06(9.08e+05)$	$4.63e+05(3.95e+05)+$
F_6	$1.09e+02(1.01e+02)$	$4.98e+01(6.21e+01)+$	$1.48e+02(9.08e+01)$	$1.57e+02(8.34e+01)\approx$	$7.32e+02(9.80e+01)$	$1.96e+01(9.12e+00)+$
F_7	$1.71e+02(1.12e+02)$	$1.60e+02(1.39e+02)\approx$	$2.25e+02(1.62e+02)$	$1.50e+02(1.20e+02)+$	$3.46e+05(2.32e+05)$	$8.16e+04(7.08e+04)+$
F_8	$9.77e+01(1.62e+01)$	$9.35e+01(2.67e+01)\approx$	$9.86e+01(1.55e+01)$	$9.30e+01(2.79e+01)+$	$1.35e+03(4.62e+02)$	$1.08e+02(5.44e+01)+$
F_9	$3.23e+02(7.45e+01)$	$3.08e+02(8.41e+01)+$	$3.06e+02(9.02e+01)$	$2.88e+02(1.02e+02)\approx$	$3.81e+02(2.63e+00)$	$3.50e+02(4.30e+00)+$
F_{10}	$4.29e+02(2.29e+01)$	$4.25e+02(2.32e+01)\approx$	$4.14e+02(2.20e+01)$	$4.29e+02(2.22e+01)-$	$4.59e+02(4.25e+00)$	$4.47e+02(6.81e+00)+$
Chaos wins	+	4	+	3	+	9
Original wins	–	1	–	1	–	1
Similar	$\approx$	5	$\approx$	6	$\approx$	0

Table 11
Comparison of algorithms (SOMA ATO, HFPSO, and ABC original vs chaos version) on the CEC 2015 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2015	SOMA ATO		HFPSO		ABC	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	1.90e+05(1.65e+05)	5.97e+04(5.64e+04)+	1.13e+04(8.36e+03)	4.83e+04(3.95e+04)–	7.03e+06(2.25e+06)	3.26e+06(5.16e+06)+
F_2	6.85e+03(7.10e+03)	4.52e+02(8.12e+02)+	6.25e+03(7.88e+03)	6.40e+03(7.74e+03) ≈	4.89e+03(4.05e+03)	1.85e+04(2.28e+04)–
F_3	2.01e+01(3.55e−02)	2.01e+01(2.75e−02)–	2.00e+01(2.05e−03)	2.00e+01(1.04e−02)–	2.03e+01(6.62e−02)	2.03e+01(8.17e−02) ≈
F_4	6.65e+00(1.85e+00)	5.32e+00(1.82e+00)+	1.45e+01(5.35e+00)	8.63e+00(3.67e+00)+	2.77e+01(3.79e+00)	1.77e+01(7.87e+00)+
F_5	2.34e+02(1.08e+02)	2.01e+02(1.19e+02) ≈	3.68e+02(1.93e+02)	2.82e+02(1.47e+02) ≈	1.41e+03(1.46e+02)	8.90e+02(3.75e+02)+
F_6	1.68e+03(1.73e+03)	1.95e+02(2.21e+02)+	2.83e+03(2.58e+03)	2.39e+03(2.81e+03) ≈	2.67e+04(1.48e+04)	6.33e+03(9.32e+03)+
F_7	4.43e−01(2.65e−01)	4.18e−01(3.68e−01)+	1.65e+00(1.00e+00)	1.46e+00(9.74e−01) ≈	2.52e+00(1.88e−01)	2.61e+00(6.50e−01) ≈
F_8	2.17e+03(3.50e+03)	6.93e+01(7.73e+01)+	1.40e+03(1.46e+03)	8.89e+02(1.02e+03) ≈	9.31e+03(5.80e+03)	7.12e+03(7.79e+03)+
F_9	1.00e+02(3.75e−02)	1.00e+02(3.94e−02)+	1.03e+02(1.67e+01)	1.00e+02(3.65e−01) ≈	1.00e+02(4.34e−02)	1.00e+02(6.12e−02) ≈
F_{10}	1.26e+03(1.40e+03)	3.68e+02(3.31e+02)+	9.87e+02(8.14e+02)	7.50e+02(6.28e+02)+	1.51e+04(6.93e+03)	6.42e+03(8.93e+03)+
F_{11}	6.47e+01(1.15e+02)	1.32e+02(1.48e+02) ≈	2.87e+02(7.52e+01)	2.78e+02(1.12e+02) ≈	3.31e+02(2.91e+01)	3.93e+02(9.57e+01)–
F_{12}	1.03e+02(4.90e−01)	1.02e+02(4.85e−01)+	1.02e+02(8.11e−01)	1.02e+02(8.46e−01) ≈	1.04e+02(6.61e−01)	1.03e+02(1.08e+00)+
F_{13}	2.95e+01(1.56e+00)	2.72e+01(2.16e+00)+	3.98e+01(4.43e+00)	3.77e+01(5.66e+00) ≈	3.90e+01(1.38e+00)	3.58e+01(3.64e+00)+
F_{14}	2.40e+03(1.15e+03)	2.63e+03(9.34e+02) ≈	4.29e+03(3.73e+03)	4.42e+03(3.51e+03) ≈	3.71e+02(1.65e+01)	3.54e+02(5.64e+01)+
F_{15}	1.00e+02(9.37e−06)	1.00e+02(0.00e+00)+	1.00e+02(0.00e+00)	1.00e+02(0.00e+00) ≈	1.00e+02(0.00e+00)	1.00e+02(0.00e+00) ≈
Chaos wins	+	11	+	2	+	9
Original wins	–	1	–	2	–	2
Similar	≈	3	≈	11	≈	4

Table 12
Comparison of algorithms (GWO, WOA, and ACOR original vs chaos version) on the CEC 2015 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2015	GWO		WOA		ACOR	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	3.41e+06(3.15e+06)	2.58e+06(2.78e+06) ≈	4.43e+06(2.47e+06)	4.05e+06(2.78e+06) ≈	5.73e+06(3.29e+06)	2.96e+05(9.64e+04)+
F_2	4.61e+06(5.40e+06)	6.01e+06(5.87e+06) ≈	4.86e+05(8.11e+05)	9.45e+04(7.30e+04)+	5.21e+03(5.60e+03)	6.51e+03(4.47e+03)−
F_3	2.04e+01(7.64e−02)	2.04e+01(7.07e−02) ≈	2.01e+01(1.07e−01)	2.01e+01(9.44e−02) ≈	2.04e+01(6.36e−02)	2.04e+01(8.54e−02) ≈
F_4	1.47e+01(6.48e+00)	1.40e+01(5.09e+00) ≈	3.98e+01(1.65e+01)	3.89e+01(1.43e+01) ≈	2.85e+01(3.64e+00)	1.84e+01(7.94e+00)+
F_5	4.70e+02(2.42e+02)	4.72e+02(2.91e+02) ≈	9.47e+02(2.89e+02)	9.04e+02(3.05e+02) ≈	1.64e+03(1.68e+02)	1.30e+03(1.57e+02)+
F_6	1.70e+04(2.02e+04)	1.36e+04(1.34e+04)+	2.66e+05(3.27e+05)	2.82e+05(3.38e+05) ≈	9.30e+03(1.18e+04)	5.84e+02(8.71e+02)+
F_7	2.43e+00(2.18e+00)	2.44e+00(1.09e+00) ≈	5.61e+00(1.38e+00)	5.12e+00(1.14e+00) ≈	3.19e+00(2.81e−01)	2.57e+00(3.10e−01)+
F_8	3.47e+04(2.33e+05)	2.02e+03(1.19e+03) ≈	5.63e+03(4.74e+03)	6.42e+03(5.31e+03) ≈	5.49e+03(3.93e+03)	6.27e+02(8.36e+02)+
F_9	1.00e+02(1.50e−01)	1.00e+02(1.29e−01) ≈	1.01e+02(2.35e−01)	1.01e+02(2.62e−01) ≈	1.00e+02(4.40e−02)	1.00e+02(4.43e−02)+
F_{10}	3.31e+03(4.49e+03)	3.03e+03(3.30e+03) ≈	8.15e+03(1.07e+04)	9.21e+03(8.12e+03) ≈	2.21e+04(1.21e+04)	1.45e+03(1.31e+03)+
F_{11}	3.00e+02(8.68e+01)	2.83e+02(1.34e+02) ≈	3.25e+02(1.05e+02)	3.03e+02(3.91e+01) ≈	5.69e+02(8.49e+01)	3.89e+02(1.19e+02)+
F_{12}	1.02e+02(7.92e−01)	1.02e+02(8.04e−01) ≈	1.08e+02(2.60e+00)	1.08e+02(3.29e+00) ≈	1.04e+02(8.60e+00)	1.02e+02(3.17e−01)+
F_{13}	3.15e+01(4.43e+00)	3.09e+01(3.93e+00) ≈	4.14e+01(3.84e+00)	4.00e+01(3.22e+00)+	4.15e+01(3.19e+00)	3.21e+01(3.51e+00)+
F_{14}	5.91e+03(2.48e+03)	5.81e+03(2.70e+03) ≈	6.46e+03(1.28e+03)	6.34e+03(1.47e+03)+	3.50e+02(1.08e+01)	7.27e+02(1.47e+03)−
F_{15}	1.07e+02(4.99e+00)	1.09e+02(5.45e+00)−	1.01e+02(5.91e−01)	1.00e+02(1.06e−01)+	1.00e+02(0.00e+00)	1.00e+02(0.00e+00) ≈
Chaos wins	+	1	+	4	+	11
Original wins	−	1	−	0	−	2
Similar	≈	13	≈	11	≈	2

Table 13
Comparison of algorithms (FA, PSO, and CA original vs chaos version) on the CEC 2015 benchmark suite (10 dimensions, 51 runs, full precision, WRT at 5%).

CEC 2015	FA		PSO		CA	
Function	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)	Original Mean (Std Dev)	Chaos Mean (Std Dev)
F_1	9.95e+04(1.16e+05)	1.61e+05(1.97e+05) ≈	1.06e+04(9.46e+03)	7.25e+04(4.59e+04)–	1.35e+08(7.17e+07)	5.87e+07(2.32e+07)+
F_2	6.09e+03(8.00e+03)	1.28e+04(1.39e+04) ≈	6.67e+03(7.16e+03)	6.75e+03(7.44e+03) ≈	1.01e+04(1.29e+04)	2.70e+07(9.68e+07)–
F_3	1.91e+01(4.83e+00)	1.97e+01(2.82e+00)–	2.00e+01(2.64e–02)	2.02e+01(1.35e–01)–	2.03e+01(7.46e–02)	2.03e+01(7.27e–02) ≈
F_4	1.31e+01(4.94e+00)	8.25e+00(4.14e+00)+	1.58e+01(7.76e+00)	1.18e+01(5.16e+00)+	4.18e+01(5.13e+00)	1.05e+01(5.89e+00)+
F_5	5.87e+02(2.89e+02)	2.45e+02(1.85e+02)+	5.27e+02(2.02e+02)	3.89e+02(2.12e+02)+	2.10e+03(1.46e+02)	6.71e+02(3.39e+02)+
F_6	2.59e+03(2.34e+03)	4.32e+03(3.16e+03)–	1.73e+03(2.06e+03)	1.20e+03(9.40e+02) ≈	1.74e+06(1.44e+06)	7.15e+04(9.78e+04)+
F_7	1.01e+00(4.68e–01)	1.21e+00(4.91e–01)–	2.07e+00(9.91e–01)	1.19e+00(8.60e–01)+	3.23e+00(2.44e–01)	2.34e+00(2.70e–01)+
F_8	1.64e+03(2.66e+03)	7.09e+02(8.50e+02)+	1.46e+03(1.73e+03)	1.15e+03(1.40e+03) ≈	3.30e+05(3.26e+05)	5.44e+04(8.06e+04)+
F_9	1.00e+02(4.98e–02)	1.00e+02(4.56e–02) ≈	1.00e+02(5.64e–02)	1.00e+02(3.96e–02)+	1.00e+02(5.28e–02)	1.00e+02(4.30e–02)–
F_{10}	1.21e+03(1.54e+03)	8.39e+02(1.15e+03) ≈	9.37e+02(7.30e+02)	7.52e+02(4.76e+02) ≈	4.51e+05(3.54e+05)	1.11e+05(1.42e+05)+
F_{11}	2.71e+02(8.93e+01)	2.83e+02(7.08e+01) ≈	2.77e+02(8.05e+01)	2.71e+02(8.92e+01) ≈	4.94e+02(9.63e+01)	4.09e+02(5.23e+01)+
F_{12}	1.02e+02(4.37e–01)	1.02e+02(3.67e–01) ≈	1.02e+02(7.13e–01)	1.02e+02(5.42e–01) ≈	1.40e+02(2.70e+01)	1.03e+02(5.54e+00)+
F_{13}	3.26e+01(4.32e+00)	2.95e+01(2.40e+00)+	3.50e+01(3.82e+00)	3.29e+01(4.44e+00)+	4.48e+01(7.75e–01)	3.83e+01(1.80e+00)+
F_{14}	5.12e+03(2.31e+03)	5.12e+03(2.10e+03) ≈	2.61e+03(2.69e+03)	2.99e+03(2.97e+03) ≈	2.87e+03(1.27e+03)	5.23e+02(6.05e+01)+
F_{15}	1.00e+02(2.88e–04)	1.00e+02(3.01e–03)–	1.00e+02(0.00e+00)	1.00e+02(0.00e+00) ≈	1.00e+02(0.00e+00)	1.01e+02(2.77e+00)–
Chaos wins	+	4	+	5	+	11
Original wins	–	4	–	2	–	3
Similar	≈	7	≈	8	≈	1

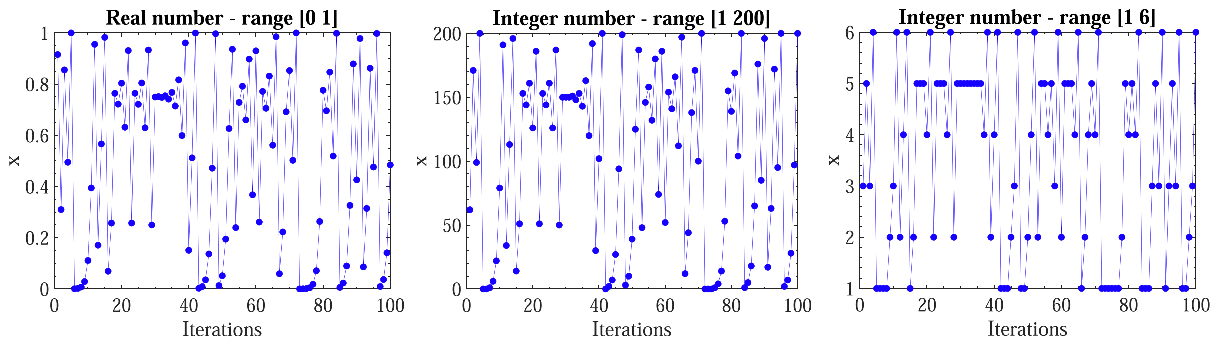

Fig. 14. The problem of integer number sequences.

This raises several interesting questions that deserve further research. Questions such as how complex and how periodic a given numerical series must be for the algorithm to work properly, whether there is a link between the chaotic mode and the nonlinearity of the test function, or how these algorithms process information about their previous activities such as N steps back and forth, etc.

Another open question, which follows from our results, is how the ability of the algorithm to find the optimal solution depends on the correlation between individual increments of time series-generated sequences (pseudo-random or chaotic). It is clear that a significant difference between randomness and chaos, from our experiment point of view, are the underlying correlations between sections of the sequences. While sections of random sequences are by definition uncorrelated, in the case of chaos short-term correlations between an initial state and subsequent states are present, see, for example, a logistic Eq. 1 with $A = 4$. How these different correlations between sections of sequences are related to algorithm performance is an open question, partly already discussed in [72–74] for example.

Based on the repeatability and accuracy of the results obtained in these experiments, it is believed that this area still offers a wide range of interesting questions and issues for further research. We also expect that the evolutionary computing community should begin to understand these algorithms as dynamic systems, for which there already exists productive mathematical apparatus dealing with their stability, control, and many other aspects. We firmly believe that applying this pre-existing mathematical apparatus to evolutionary and swarm intelligence algorithms would help shed light on the many unanswered questions that exist in the algorithm community today.

9. Conclusion

In this paper, a well-known logistic equation was used as a chaos generator, which produced deterministic chaos, as it has been verified and proven many times. The behavior of this equation was generated with varying degrees of calculation accuracy/precision. This accuracy affected whether the chaotic series was theoretically infinite and therefore unrepeatable began to repeat after N iterations. The resulting sequences of various lengths, i.e. infinitely long chaotic series but also a series of repeated partially chaotic series (so it was a kind of pseudo-chaotic periodic processes), were then used instead of pseudo-random numbers in all considered evolutionary and swarm intelligence algorithms. In other words, whenever pseudo-random numbers were needed, a chaotic series or periodic series (chaos-based) numbers were used.

The results obtained from all the experiments described in section *Results* in Figs. 12,13, and Table 4–13 clearly show the interesting positive impact of the use of N periodic series on the operation of the algorithm. Results do not need detailed comment - as clearly visible from the attached figures and tables. These results then generate the questions mentioned in section *Open Questions and Future Research*.

In our experiments, a set of algorithms were used, which, in our opinion, is sufficiently representative to demonstrate the basic idea of the paper. Of course, this does not mean that the simulations and results demonstrated in this paper cannot be extended to other algorithms such as [75–80], and many others. This leaves open the possibility for further research in this direction.

Thus, our experiments have regularly shown that chaotic series, and even periodic sequences obtained by reducing the accuracy of these series, certainly increases the performance of evolutionary algorithms. This leads to the striking fact that the algorithm performance increases even if random-like processes are not used, which contradicts a generally accepted dogma. We believe that it is an interesting research area that combines computer science in terms of algorithm study, theoretical cybernetics and control theory.

CRediT authorship contribution statement

Ivan Zelinka: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Quoc Bao Diep:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Václav Snášel:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Swagatam Das:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Giacomo Innocenti:** Conceptualization, Methodology, Software, Validation, Formal analisys, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Alberto Tesi:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Fabio Schoen:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing. **Nikolai V. Kuznetsov:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The following grants are acknowledged for the financial support provided for this research: grant of SGS No. SGS SP2021/72, VSB-Technical University of Ostrava, Czech Republic, grant Pure ID 75207094 of St.Petersburg State University, Russia. The research leading to the published results was also supported by the Ministry of the Interior of the Czech Republic under grant ID VJ01010008 within the project Network Cybersecurity in Post-Quantum Era.

References

[1] J. Barrow-Green, Poincaré and the three body problem, Am. Math. Soc. 11 (1997).
[2] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.
[3] K.-Y. Hsiao, C.-Y. Wu, Y.-T. Huang, Fluid mixing in a microchannel with longitudinal vortex generators, Chem. Eng. J. 235 (2014) 27–36.
[4] C.-C. Che, M.-C. Tian, G.-M. Zhang, X.-L. Leng, Experimental study by piv of swirling flow induced by trapezoid-winglets, J. Hydrodyn. 25 (6) (2013) 919–928.
[5] B. Seifert, D. Adamski, C. Uhl, Analytic quantification of shilnikov chaos in epileptic eeg data, Front. Appl. Math. Stat. 4 (2018) 57.
[6] D. Kozłowski, Method in the chaos–a step-by-step approach to ecg interpretation, Eur. J. Transl. Clin. Med. 1 (1) (2018) 76–92.
[7] V. Nenadovic, R. Whitney, J. Boulet, M.A. Cortez, Hypsarrhythmia in epileptic spasms: synchrony in chaos, Seizure 58 (2018) 55–61.
[8] C. Lin, T. Ng, A. Skaff, An experimental study of wing tips for wing performance improvement, 14th Applied Aerodynamics Conference (1996) 2413.
[9] A.H. Wright, A. Agapie, Cyclic and chaotic behavior in genetic algorithms, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation Citeseer, 2001, pp. 718–724.
[10] I. Zelinka, S. Celikovský, H. Richter, G. Chen, Evolutionary algorithms and chaotic systems, vol. 267, Springer, 2010.
[11] R. Lozi, Emergence of randomness from chaos, Int. J. Bifurcation Chaos 22 (02) (2012) 1250021, https://doi.org/10.1142/S0218127412500216.
[12] K. Persohn, R. Povinelli, Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation, Chaos Solitons Fractals 45 (3) (2012) 238–245, https://doi.org/10.1016/j.chaos.2011.12.006, URL:https://www.sciencedirect.com/science/article/pii/S0960077911002384.
[13] X.-Y. Wang, X. Qin, A new pseudo-random number generator based on cml and chaotic iteration, Nonlinear Dyn. 70 (2) (2012) 1589–1592, https://doi.org/10.1007/s11071-012-0558-0, URL:https://link.springer.com/article/10.1007/s11071-012-0558-0.
[14] N.K. Pareek, V. Patidar, K.K. Sud, A random bit generator using chaotic maps, Network Secur. 10 (1) (2010) 32–38.
[15] X.-Y. Wang, L. Yang, Design of pseudo-random bit generator based on chaotic maps, Int. J. Mod. Phys. B 26 (32) (2012) 1250208, https://doi.org/10.1142/S0217979212502086, URL:https://doi.org/10.1142/S0217979212502086.
[16] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, A. Rizzo, Does chaos work better than noise?, IEEE Circuits Syst Mag. 2 (3) (2002) 4–19, https://doi.org/10.1109/MCAS.2002.1167624.
[17] R. Caponetto, L. Fortuna, S. Fazzino, M. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms, IEEE Trans. Evol. Comput. 7 (3) (2003) 289–304, https://doi.org/10.1109/TEVC.2003.810069.
[18] H. Hu, L. Liu, N. Ding, Pseudorandom sequence generator based on the chen chaotic system, Comput. Phys. Commun. 184 (3) (2013) 765–768, https://doi.org/10.1016/j.cpc.2012.11.017, URL:https://www.sciencedirect.com/science/article/pii/S0010465512003931.
[19] A. Pluchino, A. Rapisarda, C. Tsallis, Noise, synchrony, and correlations at the edge of chaos, Phys. Rev. E 87 (2013), https://doi.org/10.1103/PhysRevE.87.022910, URL:https://link.aps.org/doi/10.1103/PhysRevE.87.022910 022910.
[20] M. Pluhacek, R. Senkerik, D. Davendra, K. Zuzana, I. Zelinka, On the behavior and performance of chaos driven pso algorithm with inertia weight, Comput. Math. Appl. 66(2) (2013) 122–134, nostradamus 2012. doi:10.1016/j.camwa.2013.01.016. URL:https://www.sciencedirect.com/science/article/pii/S0898122113000333..
[21] M. Pluháček, V. Budíková, R. Šenkeřík, Z. Oplatková, I. Zelinka, On the performance of enhanced pso algorithm with lozi chaotic map-an initial study, MENDEL (2012).
[22] W.-C. Hong, Y. Dong, W.Y. Zhang, L.-Y. Chen, B.K. Panigrahi, Cyclic electric load forecasting by seasonal svr with chaotic genetic algorithm, Int. J. Electrical Power Energy Syst. 44 (1) (2013) 604–614, https://doi.org/10.1016/j.ijepes.2012.08.010, URL:https://www.sciencedirect.com/science/article/pii/S0142061512004462.
[23] R. Senkerik, D. Davendra, I. Zelinka, Z. Oplatkova, M. Pluhacek, Optimization of the batch reactor by means of chaos driven differential evolution, in: V. Snášel, A. Abraham, E.S. Corchado (Eds.), Soft Computing Models in Industrial and Environmental Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 93–102. doi:10.1007/978-3-642-32922-7_10. URL:https://link.springer.com/chapter/10.1007/978-3-642-32922-7_10.
[24] D. Davendra, I. Zelinka, R. Senkerik, Chaos driven evolutionary algorithms for the task of pid control, Comput. Math. Appl. 60(4) (2010) 1088–1104, pCO' 2010. doi:10.1016/j.camwa.2010.03.066. URL:https://www.sciencedirect.com/science/article/pii/S0898122110002567..
[25] M. El-Shorbagy, A. Mousa, S. Nasr, A chaos-based evolutionary algorithm for general nonlinear programming problems, Chaos Solitons Fractals 85 (2016) 8–21, https://doi.org/10.1016/j.chaos.2016.01.007, URL:https://www.sciencedirect.com/science/article/pii/S0960077916000163.
[26] A. Viktorin, M. Pluhacek, R. Senkerik, Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on cec2014 benchmark set, in, IEEE Congress on Evolutionary Computation (CEC) 2016 (2016) 4797–4803, https://doi.org/10.1109/CEC.2016.7744404.

[27] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Software 95 (2016) 51–67.
[28] H. Chen, W. Li, X. Yang, A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems, Expert Syst. Appl. 158 (2020), https://doi.org/10.1016/j.eswa.2020.113612, URL:https://www.sciencedirect.com/science/article/pii/S095741742030436X 113612.
[29] S. Talatahari, M. Azizi, Optimization of constrained mathematical and engineering design problems using chaos game optimization, Comput. Ind. Eng. 145 (2020), https://doi.org/10.1016/j.cie.2020.106560, URL:https://www.sciencedirect.com/science/article/pii/S0360835220302941 106560.
[30] B. Xing, W.-J. Gao, Invasive weed optimization algorithm, in: Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms, Springer, 2014, pp. 177–181..
[31] M. Misaghi, M. Yaghoobi, Improved invasive weed optimization algorithm (IWO) based on chaos theory for optimal design of PID controller, J. Comput. Design Eng. 6 (3) (2019) 284–295, https://doi.org/10.1016/j.jcde.2019.01.001, URL:https://doi.org/10.1016/j.jcde.2019.01.001.
[32] Z. Song, S. Gao, Y. Yu, J. Sun, Y. Todo, Multiple chaos embedded gravitational search algorithm, IEICE Trans. Inform. Syst. 100 (4) (2017) 888–900, https://doi.org/10.1587/transinf.2016EDP7512.
[33] S. Talatahari, M. Azizi, Chaos game optimization: a novel metaheuristic algorithm, Artif. Intell. Rev. 54 (2) (2021) 917–1004, https://doi.org/10.1007/s10462-020-09867-w, URL:https://link.springer.com/article/10.1007/s10462-020-09867-w.
[34] A. Mozaffari, M. Emami, A. Fathi, A comprehensive investigation into the performance, robustness, scalability and convergence of chaos-enhanced evolutionary algorithms with boundary constraints, Artif. Intell. Rev. 52 (4) (2019) 2319–2380, https://doi.org/10.1007/s10462-018-9616-4, URL: https://link.springer.com/article/10.1007/s10462-018-9616-4.
[35] S. Ahmed, M. Mafarja, H. Faris, I. Aljarah, Feature selection using salp swarm algorithm with chaos, in: Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics Swarm Intelligence, ISMSI '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 65–69. doi:10.1145/3206185.3206198. URL:https://doi.org/10.1145/3206185.3206198..
[36] S. Hinojosa, D. Oliva, E. Cuevas, G. Pajares, O. Avalos, J. Gálvez, Improving multi-criterion optimization with chaos: a novel multi-objective chaotic crow search algorithm, Neural Comput. Appl. 29 (8) (2018) 319–335, https://doi.org/10.1007/s00521-017-3251-x, URL:https://link.springer.com/article/10.1007/s00521-017-3251-x.
[37] Q. Zhang, H. Chen, J. Luo, Y. Xu, C. Wu, C. Li, Chaos enhanced bacterial foraging optimization for global optimization, IEEE Access 6 (2018) 64905–64919, https://doi.org/10.1109/ACCESS.2018.2876996.
[38] A.M. Anter, M. Ali, Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems, Soft. Comput. 24 (3) (2020) 1565–1584, https://doi.org/10.1007/s00500-019-03988-3, URL:https://link.springer.com/article/10.1007/s00500-019-03988-3.
[39] I. Zelinka, S. Celikovský, H. Richter, G. Chen (Eds.), Evolutionary Algorithms and Chaotic Systems, Vol. 267 of Studies in Computational Intelligence, Springer, 2010. URL:http://dblp.uni-trier.de/db/series/sci/sci267.html..
[40] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optimiz. 11 (4) (1997) 341–359, https://doi.org/10.1023/A:1008202821328, URL:https://link.springer.com/article/10.1023/A:1008202821328.
[41] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, Eur. J. Oper. Res. 185 (3) (2008) 1155–1173, https://doi.org/10.1016/j.ejor.2006.06.046, URL:http://www.sciencedirect.com/science/article/pii/S0377221706006333.
[42] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, https://doi.org/10.1016/j.advengsoft.2016.01.008, URL: http://www.sciencedirect.com/science/article/pii/S0965997816300163.
[43] Ibrahim Berkan Aydilek, A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems, Appl. Soft Comput. 66 (2018) 232–249, https://doi.org/10.1016/j.asoc.2018.02.025, URL:http://www.sciencedirect.com/science/article/pii/S156849461830084X.
[44] I. Zelinka, SOMA —Self-Organizing Migrating Algorithm, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 167–217, https://doi.org/10.1007/978-3-540-39930-8_7, URL:https://doi.org/10.1007/978-3-540-39930-8_7.
[45] T. Back, B. Fogel, M.Z., Handbook of evolutionary computation, institute of physics, Institute of Physics, London..
[46] H. Schwefel, Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, ISR, vol. 26, Birkhaeuser, Basel/Stuttgart, 1977.
[47] M. Clerc, Particle swarm optimization, ISTE, London, Newport Beach, 2006.
[48] J. Kennedy, R. Eberhart, Swarm intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
[49] J.R. Koza, D. Andre, F.H. Bennett, M.A. Keane, Genetic Programming III: Darwinian Invention and Problem Solving, first ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
[50] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (abc) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57, https://doi.org/10.1007/s10462-012-9328-0, URL:https://link.springer.com/article/10.1007/s10462-012-9328-0.
[51] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[52] J. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1992.
[53] M. Pluhacek, R. Senkerik, I. Zelinka, D. Davendra, Tuning the lozi map in chaos driven pso inspired by the multi-chaotic approach, in, Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems, Springer (2014) 79–88.
[54] P. Kroemer, I. Zelinka, V. Snášel, Behaviour of pseudo-random and chaotic sources of stochasticity in nature-inspired optimization methods, Soft. Comput. 18 (4) (2014) 619–629.
[55] R. Senkerik, I. Zelinka, M. Pluhacek, Chaos-based optimization-a review, J. Adv. Eng. Comput. 1 (1) (2017) 68–79.
[56] R. Senkerik, A. Viktorin, I. Zelinka, M. Pluhacek, T. Kadavy, Z.K. Oplatkova, V. Bhateja, S.C. Satapathy, Differential evolution and deterministic chaotic series: a detailed study, in: Mendel, vol. 24, 2018, pp. 61–68..
[57] R. Senkerik, M. Pluhacek, I. Zelinka, D. Davendra, J. Janostik, Preliminary study on the randomization and sequencing for the chaos embedded heuristic, in, in: Proceedings of the Second International Afro-European Conference for Industrial Advancement AECIA 2015 Springer, 2016, pp. 591–601.
[58] I. Zelinka, M. Chadli, D. Davendra, R. Senkerik, M. Pluhacek, J. Lampinen, Do evolutionary algorithms indeed require random numbers? extended study, Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems, Springer (2013) 61–75.
[59] I. Zelinka, R. Senkerik, M. Pluhacek, Do evolutionary algorithms indeed require randomness?, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2283–2289..
[60] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. (TOMACS) 8 (1) (1998) 3–30.
[61] J. Liang, B. Qu, P. Suganthan, Q. Chen, Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization, Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 29 (2014) (2015) 625–640.
[62] N. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, in: Technical Report, Nanyang Technological University Singapore, 2016.
[63] C. Yue, K. Price, P. Suganthan, J. Liang, M. Ali, B. Qu, N. Awad, P. Biswas, Problem definitions and evaluation criteria for the cec 2020 special session and competition on single objective bound constrained numerical optimization, Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Tech. Rep 201911..
[64] I. Zelinka, SOMA—Self-organizing Migrating Algorithm, Springer International Publishing, Cham (2016) 3–49, https://doi.org/10.1007/978-3-319-28161-2_1, URL:https://doi.org/10.1007/978-3-319-28161-2_1.
[65] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 – International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948 vol 4. doi:10.1109/ICNN.1995.488968..

[66] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International symposium on stochastic algorithms, Springer, 2009, pp. 169–178. doi:10.1007/978-3-642-04944-6_14. URL: https://link.springer.com/chapter/10.1007/978-3-642-04944-6_14..
[67] M.G.H. Omran, A novel cultural algorithm for real-parameter optimization, Int. J. Comput. Math. 93 (9) (2016) 1541–1563. arXiv:https://doi.org/10.1080/00207160.2015.1067309, doi:10.1080/00207160.2015.1067309. URL:https://doi.org/10.1080/00207160.2015.1067309..
[68] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007, URL: http://www.sciencedirect.com/science/article/pii/S0965997813001853.
[69] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18, https://doi.org/10.1016/j.swevo.2011.02.002, URL:http://www.sciencedirect.com/science/article/pii/S2210650211000034.
[70] J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm Evolut. Comput. 54 (2020), https://doi.org/10.1016/j.swevo.2020.100665, URL:http://www.sciencedirect.com/science/article/pii/S2210650219302639 100665.
[71] E. Schöll, H.G. Schuster (Eds.), Handbook of Chaos Control, Wiley-VCH, Weinheim, 2008, second completely revised and enlarged edition..
[72] R. Senkerik, A. Viktorin, T. Kadavy, M. Pluhacek, I. Zelinka, Is chaotic randomization advantageous for higher dimensional optimization problems?, in: International Conference on Artificial Intelligence and Soft Computing Springer, 2020, pp 423–434.
[73] R. Senkerik, M. Pluhacek, D. Davendra, I. Zelinka, Z.K. Oplatkova, Chaos driven evolutionary algorithm: A new approach for evolutionary optimization, Int. J. Appl. Math., Comput. Sci. Syst. Eng. 3..
[74] R. Šenkeřík, A brief overview of the synergy between metaheuristics and unconventional dynamics, Lecture notes in electrical engineering..
[75] H. Li, F. He, Y. Chen, Y. Pan, Mlfs-ccde: multi-objective large-scale feature selection by cooperative coevolutionary differential evolution, Memetic Comput. 13 (1) (2021) 1–18.
[76] J. Luo, F. He, H. Li, Y. Liang, A novel whale optimization algorithm with filtering disturbance and non-linear step, Int. J. Bio-Inspired Comput. 16 (2020) 137–148.
[77] Y. Liang, F. He, X. Zeng, 3d mesh simplification with feature preservation based on whale optimization algorithm and differential evolution, Integrated Comput. -Aided Eng. (Preprint) (2020) 1–19.
[78] Y. Chen, F. He, H. Li, D. Zhang, Y. Wu, A full migration bbo algorithm with enhanced population quality bounds for multimodal biomedical image registration, Appl. Soft Comput. 93 (2020) 106335.
[79] N. Nedjah, L.D.M. Mourelle, R.G. Morais, Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation: a survey-part ii, Int. J. Bio-Inspired Comput. 16 (4) (2020) 195–212.
[80] J. Xue, B. Shen, A novel swarm intelligence optimization approach: sparrow search algorithm, Syst. Sci. Control Eng. 8 (1) (2020) 22–34.