Chaotic attractors of discrete dynamical systems used in the core of evolutionary algorithms: state of art and perspectives

Chaotic attractors of discrete dynamical systems used in the core of evolutionary algorithms: state of art and perspectives

Ivan Zelinka[a,b] and Roman Senkerik[c]

[a]*Department of Computer Science, VSB Technical University of Ostrava, Ostrava-Poruba, Czech Republic*

[b]*IT4Innovations, VSB-Technical University of Ostrava, Ostrava-Poruba, Czech Republic*

[c]*A.I.Lab, Faculty of Applied Informatics,Tomas Bata University in Zlin, Zlín, Czech Republic*

CONTACT Roman Senkerik: senkerik@utb.cz, A.I.Lab, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam.T. G. Masaryka 5555, Zlín 760 01, Czech Republic

ABSTRACT

This paper delves into the intersection between discrete dynamical systems and bio-inspired metaheuristic algorithms. Chaos, an inherent phenomenon manifested by both continuous and discrete dynamic nonlinear systems has become an essential component of engineering design. This study examines the relationship between chaos and bio-inspired metaheuristic algorithms from two distinct angles: the presence of chaos within the realm of algorithms, and the application of bio-inspired algorithms for the identification, control, or synthesis of complex systems. Furthermore, this paper offers an in-depth exploration of the interplay between metaheuristics and complex nonlinear dynamics, including potential avenues for future research. The presented simulation studies, along with the design of objective functions for optimization and the implementation of metaheuristics, serve as a valuable foundation for subsequent experiments involving discrete nonlinear systems and complex systems with time delays. Such systems may benefit from multiparametric or multicriteria optimization approaches, paving the way for novel advancements in the field.

KEYWORDS: Deterministic chaos, metaheuristics, evolution algorithms, swarm intelligence, discrete chaotic dynamical systems, chaotic maps

1. Introduction

In modern science, difference and differential equations play an indispensable role in the mathematical apparatus. They can be used to solve many real engineering problems and define fundamental research theorems. Classical tools have long been used to solve such equations and systems of equations, but in the last several decades, computers and algorithms have come to play a very important role. The family of algorithms allowing us to solve problems based on difference equations is very rich. The metaheuristic algorithms (including evolutionary algorithms and swarm intelligence) can not only be used to optimize tasks defined by set of differential equations but also to

design their structure given the measured data from a given problem. They can also be used to control physical systems studied and described using difference or differential equations, control and synthesize chaotic systems, and solve/optimize many other complex problems.

This paper represents a summary of selected areas with an interesting intersection of both classical mathematics in the field of difference equations and their solution and modern optimization approaches that belong to the family of metaheuristics, specifically evolutionary algorithms and swarm intelligence. The main goal of this paper is only to draw attention to the existence of the intersection - fusion of these two fields and their importance. The focus here is on discrete nonlinear systems producing chaotic behaviour and how such systems can be modeled, stabilized, identified, or possibly synthesized using modern algorithmic techniques.

The organization of this paper is the following. First, a brief background on deterministic chaos, metaheuristics (evolutionary computational techniques), and the motivation that has led to many research papers linking discrete or continuous chaos and metaheuristics are provided. Selected important cases are presented, such as the analysis of chaos in metaheuristics, the evolutionary synthesis of new chaotic systems, and especially the evolutionary control of discrete chaotic maps and real systems. These technical demonstrations include a literature review, references to related publications, and a briefdiscussion. Finally, all the findings are summarized in a brief conclusion.

1.1.Deterministic chaos

One of the important moments in the twentieth century is the emergence of the 'new science' popularly known as 'Chaos.' Many researchers were involved in its creation, and this gradually emerging discipline can be traced back to the 19th century. For example, so-called Lyapunov exponents are typical examples. These exponents, usually referred to as λ, characterize the rate at which nearby trajectories in a dynamical system diverge or converge over time. It provides a measure of the sensitivity of a system to initial conditions, which is a key feature of chaotic systems. The Lyapunov exponent quantifies the exponential separation rate between two initially close trajectories in phase space as they evolve in time. If the exponent is positive, it indicates that the trajectories diverge exponentially, suggesting chaos. Conversely, if the exponent is negative, the trajectories converge exponentially, indicating stability. A zero Lyapunov exponent implies that the trajectories neither converge nor diverge significantly, resulting in a neutral or marginally stable behaviour. More details can be found in survey paper [**3**].

Another example is the celestial mechanic's field, specifically the study of the motion of three bodies in gravitational interaction [**39**]. The three-body problem is one of the fundamental demonstrations of the deterministic chaos of so-called Hamiltonian systems.

Popularization and increased interest in chaos are linked to E. N. Lorenz, who became famous for discovering the so-called 'Butterfly Effect,' which he presented in the 1960s. The butterfly effect refers to the impossibility of long-term weather forecasting (max. 2-3 days) and, thus to its chaotic nature. The so-called catastrophe theory is closely related to chaos theory, developed by the French mathematician Rene Thom in the 1960s [**52**].

In recent decades new types of chaotic systems have been discovered [**9, 53**], which according to [**55**], are so-called dual systems. In parallel with these systems, other types have been described and intensively studied, such as (dissipative) Lozi, Tinkerbell, Ikeda, Sinai, Burger and Duffing systems, or (conservative) Chirikov, Arnold, Baker, Nose-Hoover, and Henon-Heiles systems. These systems are

mainly derived from existing physical systems. Typical examples are the Lorenz system (weather), the Duffing system (steel belt in a magnetic field), and the logistic equation (biological hunter-prey system).

1.2. Evolutionary computational techniques

Optimization algorithms are a powerful tool for solving many problems in engineering practice. They are usually used where solving a problem by analytical methods is inappropriate or impracticable. 'Metaheuristics' is an established term for modern nature-inspired optimization methods, first used in 1986 [**23**]. Without any doubt, it can be said that nowadays, evolutionary algorithms (EAs) represent the most popular metaheuristics [**17**]. In computer science, EAs or so-called evolutionary computation techniques ($ECTs$) are a subfield of artificial intelligence (AI) that allows for solving complex optimization problems. $ECTs$ are inspired by Darwinian principles of natural selection as a critical mechanism of evolution and Mendel's law of heredity [**5, 21**].

The operation of evolutionary algorithms can be generally characterized as the sequence of operations (Algorithm 1). To get a clear idea of the functionality of the typical sequence of operations, it is still necessary to define the relationship between the optimization problem itself and the EA/ECT, i.e. how the algorithm obtains feedback on the quality of the solution to control the evolutionary process using specific operators. This connecting element is the objective function (or fitness function). The best solution is the global extreme - the maximum or minimum of the objective function.

Algorithm 1 Evolution algorithm

```
(Randomly) generate the initial population of solutions;
Evaluate each solution in the population;
while the termination condition is not satisfied do
    Select the solutions which are highly suitable for reproduction (parents);
    Recombine pairs of parents (crossover);
    Mutate the resulting offspring;
    Evaluate (quality of) new solutions;
    Replace inferior solutions with new solutions;
end while
Return the best solution found;
```

As already mentioned, metaheuristics often find inspiration in biological phenomena. Over time, two large groups of algorithms have emerged, distinguished by different philosophies of function (mainly due to different 'inspirations'). Thus, in the literature and in the scientific community, it is possible to find the labelling of these two important groups as 'classical EA' and 'swarm' algorithms. Classical evolutionary algorithms generally use an iterative process representing the evolution of a population. This class of algorithms is characterized by the use of special operators to drive the evolutionary process to find the ideal solution, such as the genome, mutation, crossover (recombination), and others. The classical evolutionary algorithms mainly include evolutionary strategies [**6**], genetic algorithms (GA) [**24**] and differential evolution (DE) [**11, 51**].

GAs is a popular example of a bio-inspired optimization technique that mimics the process of natural selection to find optimal or near-optimal solutions to complex problems. GAs are widely used in various domains, including optimization, machine learning, and artificial intelligence, due to their ability to effectively explore large and complex search spaces. GAs works by iteratively evolving a population of candidate solutions using simple steps. At the end of the algorithm, the solution with the highest fitness value in the final population is considered the best-found solution.

(1) $Initialization$: Randomly create a population of candidate solutions (usually represented as binary strings or other data structures).
(2) $Fitness\ Evaluation$: Assess the quality of each candidate solution in the population by calculating its fitness value using a predefined objective function. The fitness function is the normalized objective function in the interval 0-1.
(3) $Selection$: Choose a subset of the population for reproduction based on their fitness values. Higher-fitness solutions are more likely to be selected, emulating the survival of the fittest principle. Usually, the ranking of solutions or tournament system is used.
(4) $Crossover$: Generate new offspring solutions by recombining the selected parent solutions. This is typically done by exchanging parts of their genetic information, such as swapping sections of their binary strings.
(5) $Mutation$: Introduce small random changes into the offspring solutions to promote diversity in the population and prevent premature convergence to suboptimal solutions. This process is analogous to genetic mutations in biological systems. Simple binary inversion with very low probability is common.
(6) $Termination$: Repeat steps 2 to 5 for a predefined number of generations until a termination criterion is met (e.g. reaching a maximum number of generations, spending the budget of objective/fitness function evaluation, finding a solution with a satisfactory fitness value, or observing no significant improvement over several generations).

Many metaheuristics find inspiration in the collective intelligence of animal societies. These are called swarm algorithms. Some species make intelligent decisions without central control. Based on the limited knowledge of an individual, swarms achieve their optimization goals, e.g.for foraging. Modeling these events has led to the development of popular algorithms such as Particle Swarm Optimization (PSO) [**18**], Ant Colony Optimization [**14**], Self-organizing Migration Algorithm ($SOMA$) [**65**], whose successes were then followed by Artificial Bee Colony [**26**], Grey Wolf Optimizer [**35**] and many others.

Apart from the above, algorithms using inspiration in physical laws [**41**] or cognitive science [**7**] are also popular. However, it is important not to forget another important subfield of ECT, namely symbolic regression. Symbolic regression (symbolic feature identification) is an approach to search for features to analyze data sets to synthesize data-generating systems (models) fully. Symbolic regression is the process of synthesizing a final model from the basic simple functions, operators, and terminals (variables and constants) that correspond to, for example, a given data set.

John Koza first introduced the symbolic regression approach as the well-known concept of genetic programming (GP) [**27**], later followed by Conor Ryan with grammatical evolution (GE) [**44**]. There are also many other methods and areas, such as Evolutionary Hardware and Cartesian Genetic Programming [**56**]. Still, above all, we should mention the symbolic regression method, namely Analytical Programming (AP), developed by Prof. Ivan Zelinka [**69**] in 2001, which became the basis of many of the experiments presented here.

1.3. Motivation

The apparent motivation behind this research survey is to show how, and especially why, these algorithms are so important and interesting in nonlinear systems and how they can be applied in an engineering way.

As such, the issue of optimization has been an essential topic of research with various application domains for many past years. The problems of optimization have been solved by the classical mathematical apparatus, numerical methods (descent or pattern searching), techniques based on (non)linear programming (including mixed integer versions), and many more.

The computational and algorithmic complexity increases not only with the complexity of the optimized problem given by the number of parameters (arguments, dimensions) but also with the number of constraints and the domain of arguments of the optimized function (real, integer, logical, linguistic). Thus, to avoid unbearable computational demands in many cases, the need to implement complex solvers and often use the domain knowledge of a large team of experts, it is advantageous to use computational methods based on metaheuristics.

Using these methods requires only a very good knowledge of the optimized problem, often the (external) relationship between the input and output data, i.e. correctly defining the objective function whose optimization should lead to the problem's solution. Another advantage is that these algorithms, by their nature, always aim at finding the global extreme. Many of them, as population-based algorithms, provide at the end of their run a large set of different quality solutions, which may differ from each other in the value of the objective function and the position in the search space. Moreover, these algorithms are suitable for parallelization and execution on modern platforms. The disadvantage of these algorithms is that they work with stochastic operations, and there is no guarantee of optimality.

Figure 1 shows the landscape of an objective function (input/output mapping) constructed for synchronizing chaotic systems represented by a set of differential equations. Each point of this space represents one possible control action on the system through coupling parameters. As can be seen, the landscape is very complex (multimodal containing many local extremes), and finding an optimal solution representing an optimal control intervention using classical methods can be extremely difficult, if not impossible, even in this limited area. In this case, metaheuristics (evolutionary and swarm algorithms) stand out for their performance and efficiency, as they can find a good solution quickly. They can quickly explore the space and target specific areas even in such a very complex problem. The details, limitations, advantages, and disadvantages of using metaheuristics and different constructions of the objective function are discussed in the following sections devoted to different case studies linking metaheuristics and deterministic chaos. For more examples and discussions, refer to [**68**].

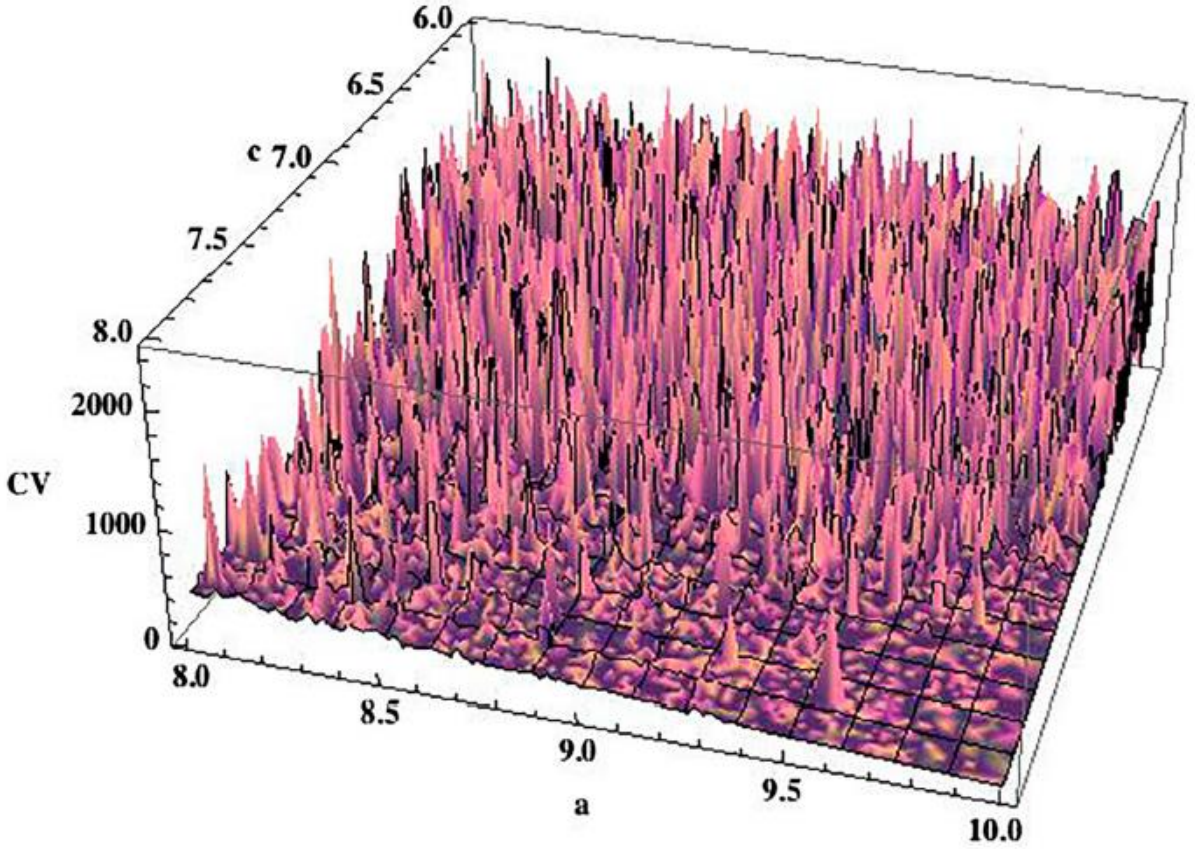

Figure 1. Objective function landscape for evolutionarily driven chaos synchronization problem. Atypical example from [**68**], where the task was to find optimal values of coupling parameters a and c. CV stands for Cost function Value.

2. Chaos in the dynamics of artificial intelligence models and metaheuristics

For many reasons like analysis of population behaviour, tuning of parameters, or simply investigating the effectiveness and applicability of evolutionary algorithms to given tasks, it is very important to know how algorithms behave under certain conditions and whether there might be hidden problems in their dynamics. If we look at evolutionary algorithms, we find that they are actually discrete dynamical systems that can also be represented by differential or difference equations. Hence, we can assume that these algorithms can externally exhibit the behaviour that is proper to classical dynamical systems modeled by standard differential equations.

The first researchers to point this out were Agapi and Wright in 2001 [**61**] in their study of genetic algorithms. They demonstrated the presence of chaotic dynamics within the algorithm itself and discussed the impact on its operation. In [**61**], models of dynamic systems of genetic algorithms are considered when the population size goes to infinity. Their work is based on the research of [**58-60**]. The elegant theory of simple genetic algorithms is based on a random heuristic search on the idea of heuristic map G [**61**]. An important research point in [**61**] is that map G includes all the dynamics of a simple genetic algorithm. Sample bifurcation diagrams are shown in Figures 2-3. Ideas about chaos existence in a simple genetic algorithm are explained in detail in 61, 68]. In both publications was shown that chaos can be observed in heuristic algorithms, which certainly does not apply only to simple genetic algorithms.

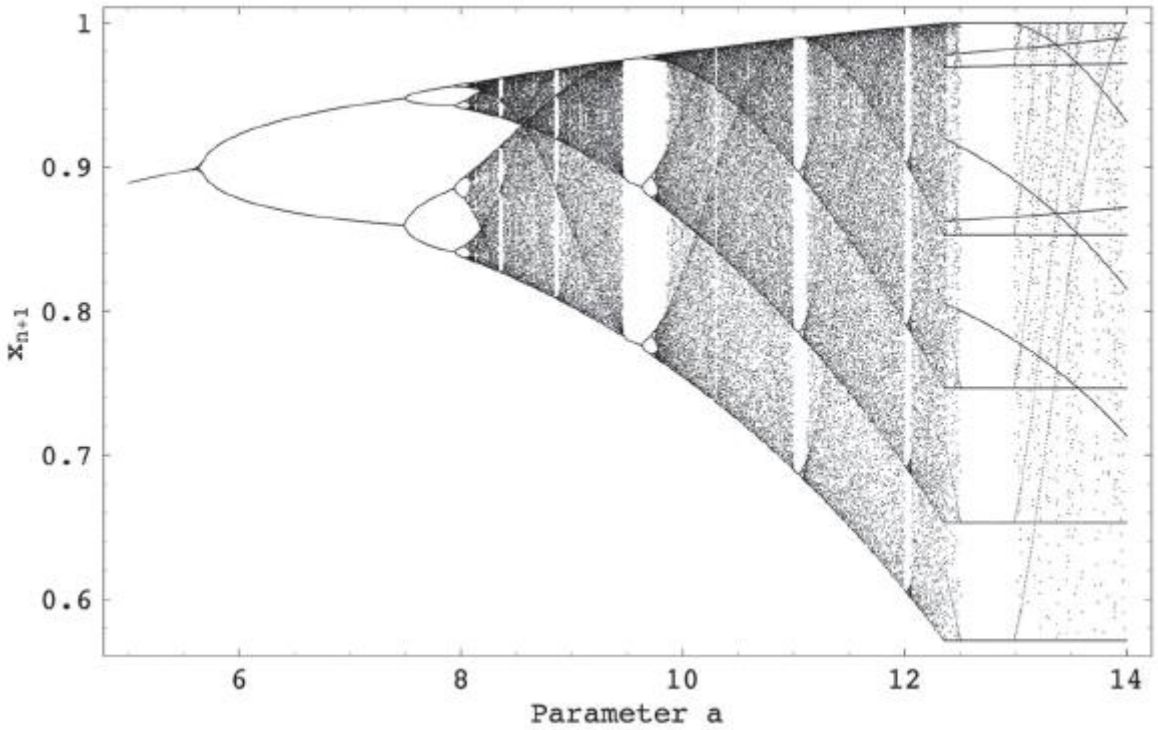

Figure 2. Bifurcation diagram of simple GA. The control parameter a belongs to the triplet of control parameters a, b, T ofheuristic map G used to model the behaviour of GA. Reconstructed accordingto [**61**].

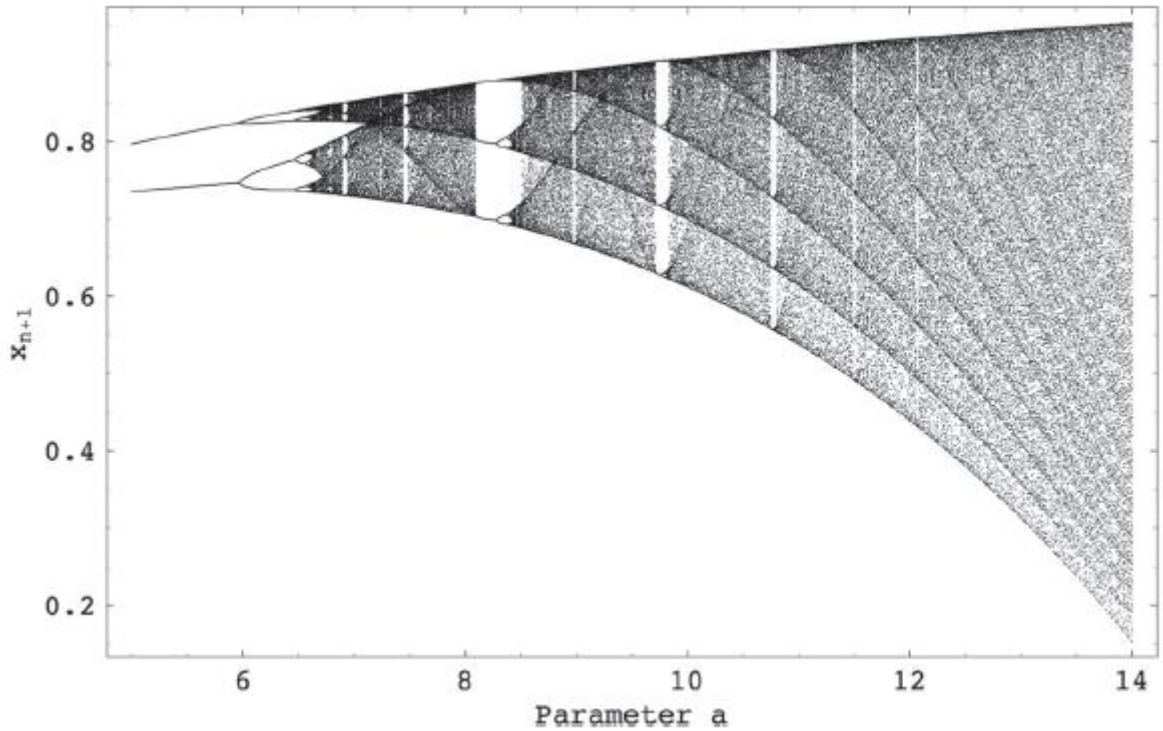

Figure 3. Bifurcation diagram of simple GA. The control parameter a belongs to the triplet of control parameters a, b, T ofheuristic map G used to model the behaviour of GA. Reconstructed accordingto [**61**].

While AI algorithms and models are typically designed to be deterministic and predictable, some AI models can exhibit chaotic behaviour under certain conditions. Recurrent Neural Networks ($RNNs$) [**33**] are a class of neural networks that can process data sequences and maintain an internal state across time steps. $RNNs$, particularly when using nonlinear activation functions, can exhibit chaotic behaviour when their internal dynamics become sensitive to small changes in input or parameters. Another AI model exhibiting chaotic features is Echo State Networks ($ESNs$) [**30**], a reservoir computing model consisting of a large, fixed, and randomly connected recurrent neural network called the reservoir.

The reservoir can exhibit rich, complex, and even chaotic dynamics, depending on the choice of connection weights and activation functions. Finally, even if it is not directly an AI model, some Cellular Automata rules can lead to chaotic behaviour, as observed in the famous Game of Life by John Conway or Rule 30 by Stephen Wolfram.

The presence of chaos in AI models, metaheuristic, is often considered a challenge to be mitigated or controlled, as it can make the model's behaviour difficult to predict, analyze and interpret. However, in some cases, controlled chaos, or the state called 'the edge of chaos' can be harnessed to improve the exploration and learning capabilities of AI models.

3. Chaos synthesis and identification

The challenging task is not only to solve differential equations and their behaviour, but also to synthesize them based on existing data. In the experiments described in this section, we point out that evolutionary techniques can also synthesize a system of equations to satisfy predefined criteria (i.e. exhibits deterministic chaos).

In recent decades, there have been introduced many different approaches to the construction of chaotic systems, or also, in other words, the synthesis of chaos. Synthesizing chaotic systems can be done through various methods, often involving mathematical models, systematicapproaches, orphysical systems designedto exhibitchaoticbehaviour [**2,10, 12, 13, 19, 54, 72**].

An original approach towards synthesizing chaotic systems based on the principles of symbolic regression is proposed in [**66**]. It is a synthesis of dynamical systems that exhibit chaotic behaviour. The complex structure synthesis was based on symbolic objects (terminals, non-terminals) $\{x, A, +, -, \times, /\}$ contained in one of the simplest chaotic systems, the logistic equation, as in (1).

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

During these experiments, many new descriptions of chaotic systems in the form of differential equations were synthesized. In the **Figures 4-8** are selected four examples of bifurcation diagrams of these artificially created chaotic systems, which demonstrate that evolution has been able to construct sets of equations to generate chaotic behaviour in specific interval. An example of the definition of such a synthesized system is given in (2) and **Figure 4**. More details can be found in [**66, 68**].

$$x_{n+1} = -\frac{A}{2x_n(-\frac{x_n^3}{A^3} - \frac{A^2}{x_n} + A)} \tag{2}$$

On the other hand, successful attempts have been made to identify the equations in the analytical description so that their behaviour fits the observed data exactly. The research, published in [**70**], is illustrated in **Figure 9** which demonstrates finding an analytical description of a system such that the system generates the same bifurcation diagram as the original for which a description was searched. The figure shows essentially two bifurcation diagrams one in black and the other in red. The black bifurcation diagram is the original data to which the mathematical description was searched and the red bifurcation diagram is the diagram generated by the found mathematical description as constructed by the evolutionary algorithms.

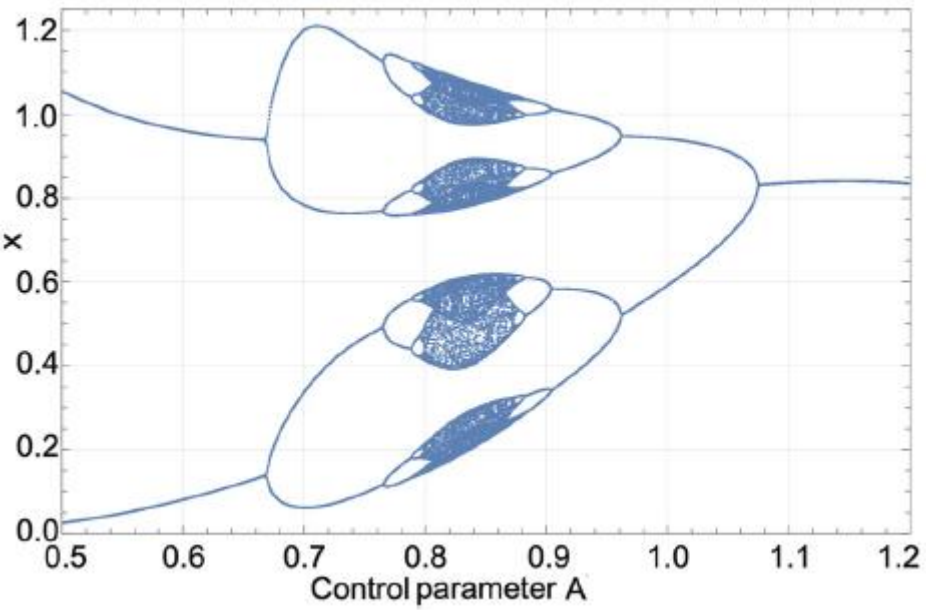

Figure 4. Bifurcation diagram of evolutionary synthesized chaotic system (2).

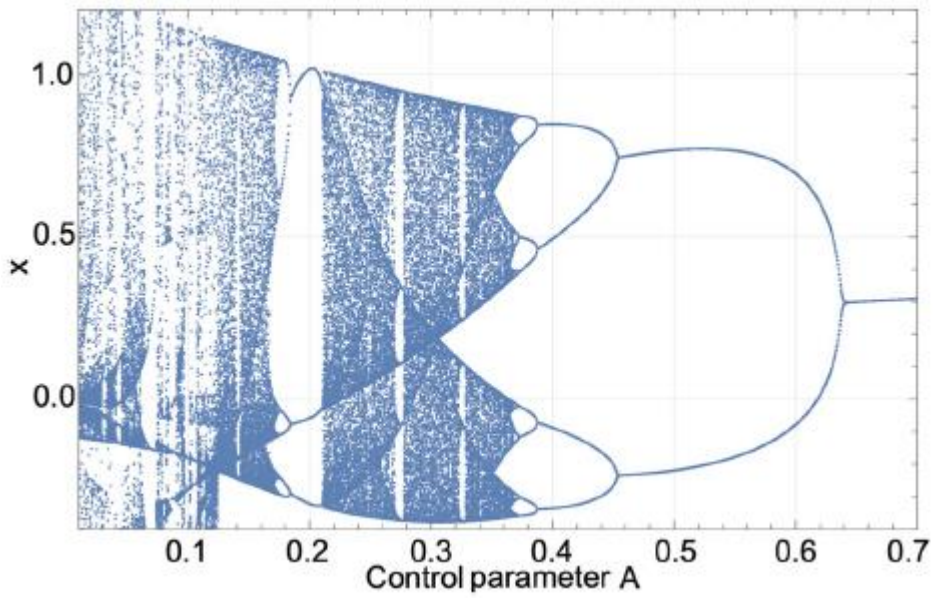

Figure 5. Bifurcation diagram of an evolutionary synthesized chaotic system, selected example from [**66,68**].

The difference between the two bifurcation diagrams is practically minimal. Finally, however, it should be noted that the above-mentioned approach of identifying chaos through bifurcation diagrams is

based on the consideration of equilibrium states and therefore works well for self-excitation of chaos, but does not allow one to effectively reveal hidden chaotic attractors (for example, in systems without equilibrium states).

4. Control

This section aims to present possible approaches to control various chaotic systems using evolutionary (metaheuristic) algorithms. The focus here is on the design of an objective function that gives feedback to the metaheuristics on the quality of the process of searching the space for possible solutions. Furthermore, the influence of chaotic dynamics and the design ofthe objective function on the shape ofthe hyperplane and thus on the possibilities and limitations of the efficiency of metaheuristic algorithms is discussed. Individual case studies are then presented in each subsection, including simple discrete chaotic maps with mention of coupled map lattices (CML) systems and an example of a real chaotic system.

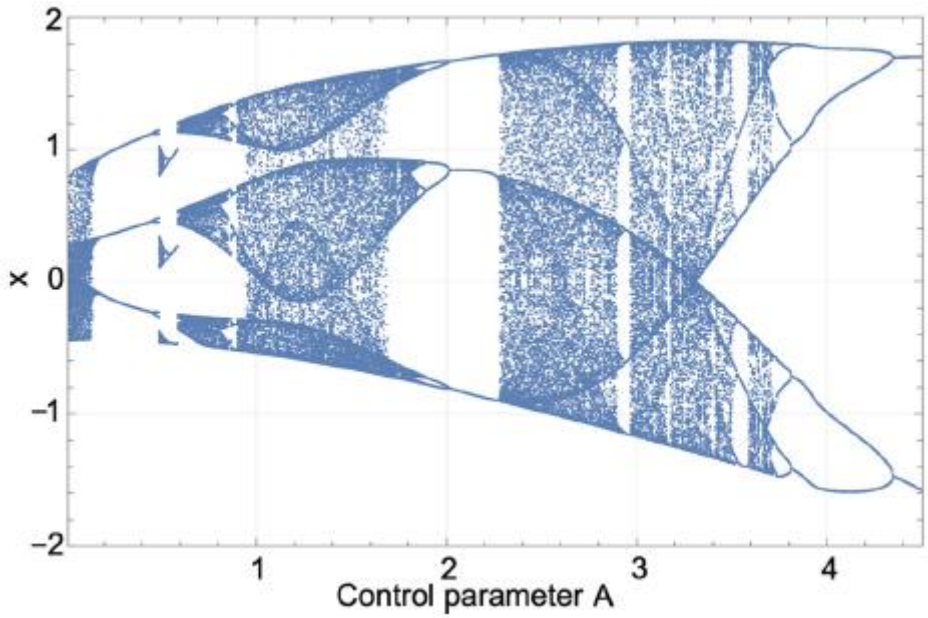

Figure 6. Bifurcation diagram of an evolutionary synthesized chaotic system, selected example from [**66,68**].

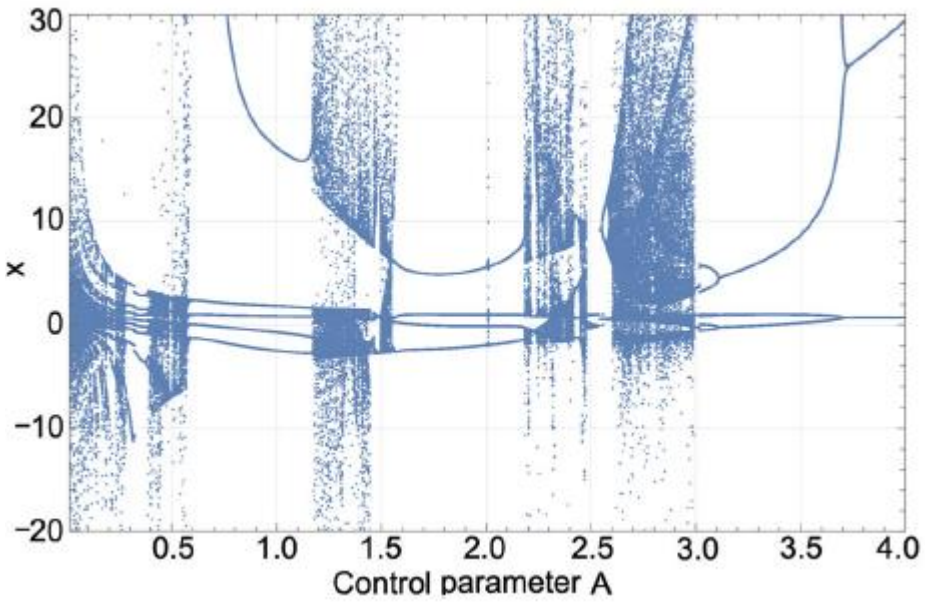

Figure 7. Bifurcation diagram of an evolutionary synthesized chaotic system, selected example from [**66, 68**].

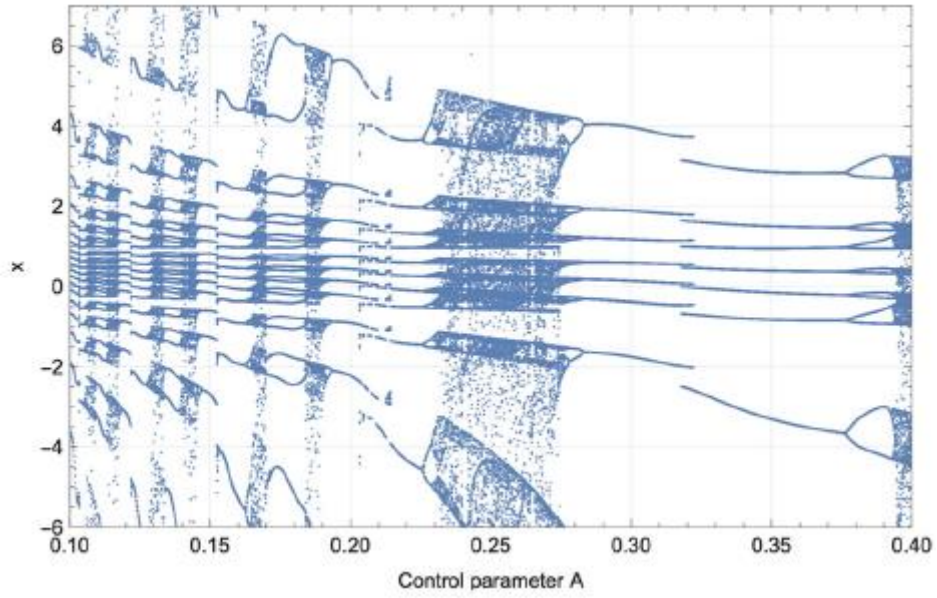

Figure 8. Bifurcation diagram of an evolutionary synthesized chaotic system, selected example from [**66,68**]. A zoom of **Figure 7**.

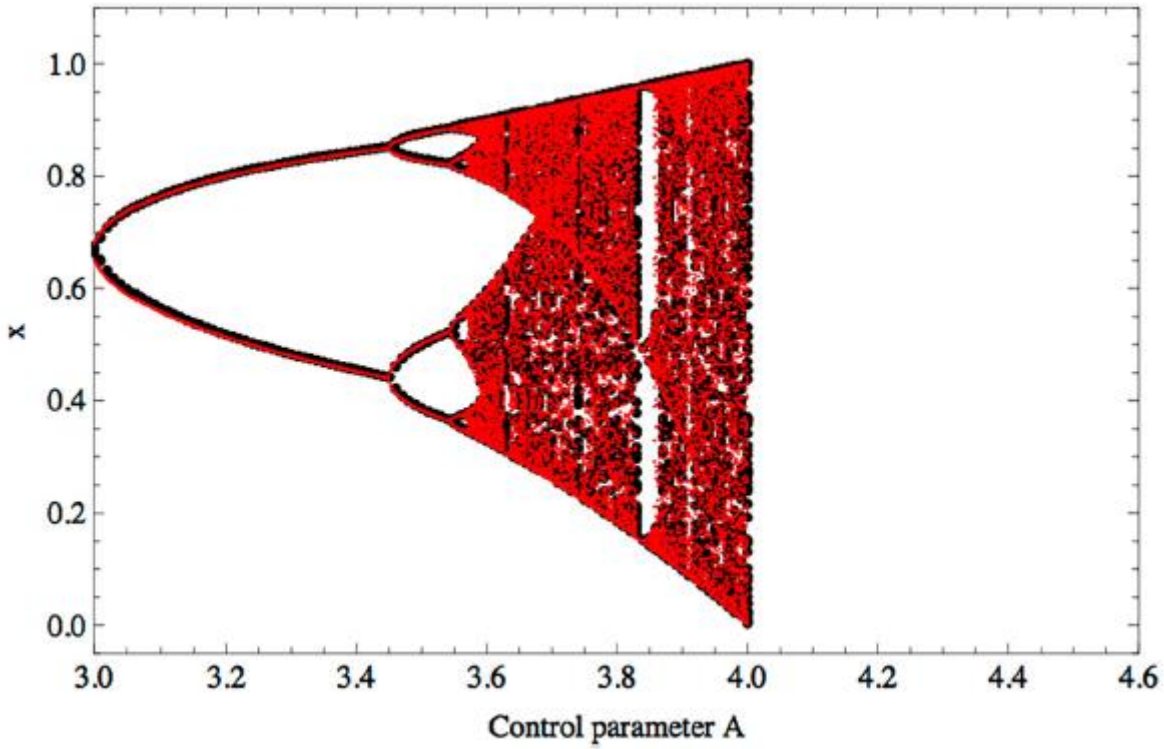

Figure 9. Bifurcation diagrams of discrete chaotic system identification. Black - original system, red - identified. Synthesis of a system based on the components of Equation (1).

4.1. Control of simple chaotic systems

Since the early 1990s, when OGY (Ott, Grebogi and Yorke) method [**38**] has been introduced, many chaos control methods have been developed like sliding mode [**62**], based on fuzzy systems [**22**] or local chaos control based on the Lyapunov approach [**42**]. Many targeting algorithms have been published with emphasis on a significant reduction of the stabilization time, simply to overcome several drawbacks of **OGY** [**20**] related to optimality and with the long initial chaotic waiting phase before the trajectory is stabilized. One of these techniques is the Pyragas' delayed feedback control method [**40**], which can be considered a 'targeting' algorithm. Despite the limitations of the Pyragas method identified in [**28**], the reason why this technique is suitable for metaheuristic-based optimization is the number of available control parameters, which have to be set otherwise by inaccurate estimation or demanding mathematical analysis. The application of evolutionary algorithms may lead to improved system behaviour and faster stabilization of the desired unstable periodic orbit (UPO) state - either a steady fixed point ($p-1$ orbit) or an arbitrarily stable periodic event ($p-2$ or higher orbit).

Many methods have also been developed to control the so-called space-time chaos that is generated, among other things, by CML systems. Control laws for CML have been derived based on a priori knowledge of the structure of CML systems, which usually remains hidden in the real world [**45**]. Methods such as external observer [**8**], evolutionary algorithms [**66**], etc. have also been used to control CMLs.

The connection between metaheuristics and deterministic chaos control has been explored and published in recent years in papers [**4, 15, 32, 43, 63, 64, 67**].

In the following two subsections, the control law is based on the above-mentioned method of Pyragas Extended Delayed Feedback - $ETDAS$. Therefore, visualizations of the objective (cost) function landscapes are shown for different combinations of $ETDAS$ method parameters.

4.2. Objective function design

Nowadays, metaheuristics optimization algorithms are known to be an effective tool for almost any optimization problem with various levels of complexity and difficulty to solve. But the quality of optimization results usually depends on the design of the objective function, especially when evolutionary algorithms are used to control the chaos. Since we are dealing with the minimization problem for parametric optimization (minimizing the cost of stabilization), we are referring to the objective function as the cost function (CF). The cost function is used to transform the search space (optimized parameters) into a geometric problem by adding an additional dimension defining the quality of the solution, and the metaheuristic optimization algorithm then tries to find the global extreme of the function (minimum or maximum) by searching the neighbourhood, cooperations between solutions, or operators such as crossover and mutation.

It is well known that deterministic chaos and chaos control techniques are very sensitive to parameter settings and initial conditions. In the case of parametric optimization, they are also extremely sensitive to the design of the cost function used, which is used for evaluating quality for a certain combination of optimized parameters for the control/synchronization method. Regarding sensitivityto initial conditions, this can be prevented in more ways than the design of the optimization cost function discussed here. The dependence of chaos on the initial data can be overcome by the requirement to have one connected global attractor. This means all trajectories, except for a set of measure zero, are attracted to one chaotic set (a self-excited attractor with some mixing properties, e.g. in the classical Lorenz system). Regarding the reported results here for the Henon attractor, it is important to avoid multistability and the presence of hidden attractors (for details, please refer to [**16**]).

There are several approaches to designing an objective/cost function to control or synchronize chaos. The most critical factor for calculating the quality of the solution is to choose appropriate metrics or indicators describing that the system has been stabilized, and then to appropriately transform their values, add any penalties and other optimality criteria (number of steps, robustness for different initial conditions, etc.) The authors in [**31**] use the $ITAE$ criterion (Integral of the Time weighted Absolute Error). The paper [**1**] uses complex definitions of cost functions implementing calculations with Jacobi matrices and spectral radius.

Below is a summary of proposals from the authors of this research survey that have been used in several published papers [**37, 46, 47**]. The various proposals include simple functions, more complex ones with multiple criteria (but still intended for single-criteria metaheuristic algorithms), and so-called black-box functions that do not require explicit prior knowledge of the periodic orbit for stabilization. A summary of the advantages and disadvantages of the various designs of the cost functions also complements the overall survey.

The proposal of the basic cost function (CF_{Simple}) is based on minimizing the area created by the sum of the differences in discrete times t between the required (target) state TS_t and the real system output (actual state) AS_t on the whole simulation interval with a total of n discrete time steps (3). This CF design is very convenient for the evolutionary searching process due to the relatively favourable CF landscape. (See **Figure 11**.).

$$CF_{Simple} = \sum_{t=0}^{n} |TS_t - AS_t| \qquad (3)$$

But this simple approach has one major drawback: the inclusion of the initial chaotic transient behaviour of the not-stabilized system in the value of the cost function. Consequently, the parameter search cannot be targeted for precise stabilization. The slight numerical difference in stabilization quality is suppressed by including the initial chaotic transient behaviour mentioned above. On the other hand, this proposal should aim the metaheuristic algorithm to search for the shortest possible overall initial chaotic behaviour, hence to time optimality. Still, it depends on the amplitude of the chaotic oscillations. Of course, this design is only suitable for stabilizing the system to a steady state ($p-1$ orbit) since the phase shift between TS_t and AS_t is not known in advance.

Different designs of the cost functions should secure the successful stabilization of either $p-1$ orbit (stable state) or any other higher periodic orbit anywise phase shifted. The so-called universal cost function is based on searching for the desired stabilized periodic orbit and thereafter, calculation of the difference between desired target state (value) and found actual periodic orbit on the short time interval - τ_s (20 iterations) from the point, where the first minimal (pre-set small) value of the difference between desired and actual system output is found (i.e. floating window for minimization - see Figure 10). This value is fixed and was set based on experimental results. It was purposely chosen so that there is no mismatch between the system's sensitivity to changes in adjustable parameters and changes in the values of the cost function. Furthermore, due to final cost values converging towards zero (in case of successful stabilization), this CF also allows the using of a stopping criterion, avoiding very time-demanding simulations. The CF_{UNI} has the form (4).

$$CF_{UNI} = p + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| \qquad (4)$$

Where:

τ_1 - the first min value of difference between TS and AS

τ_2 - the end of optimization interval ($\tau_1 + \tau_s$)

p $= 0$ if $\tau_i - \tau_2 \geq \tau_s$

$p = 10 * (\tau_i$ — t2) if Ti — $\tau_2 < \tau_s$ (i.e. late stabilization).

In order to include the time optimality of the stabilization, it was necessary to modify the definition of the universal CF_{UNI} to convert the newly defined multi-criteria problem into a single-criteria one. The simplest, but at the same time the most problematic from the point of view of motion on the search space, is that the entire CF is multiplied by the number of iterations (NI) of the first found the minimum value of the difference between the desired and the actual output of the system (the origin of the fully stabilized UPO). However, given the extreme sensitivity of the chaotic system to the control method parameter settings, this could override the emphasis on stabilization accuracy (i.e. multiplying part of the cost function by the order of tens to hundreds of iterations). At the same time, it is still

necessary to work separately with the part of the cost function for stabilization quality and the part for time optimality.

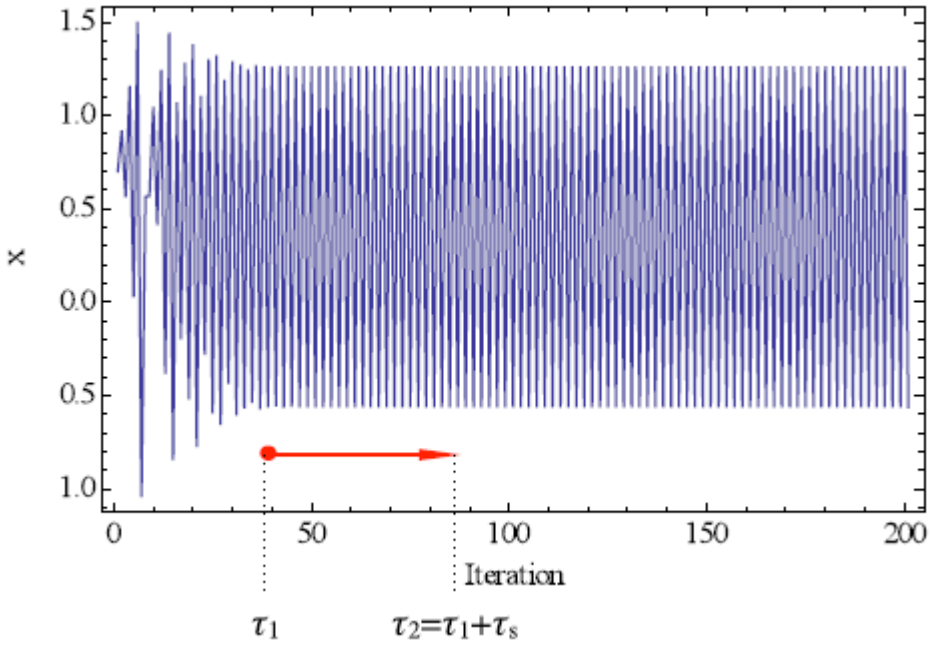

Figure 10. 'Floating window'for minimization.

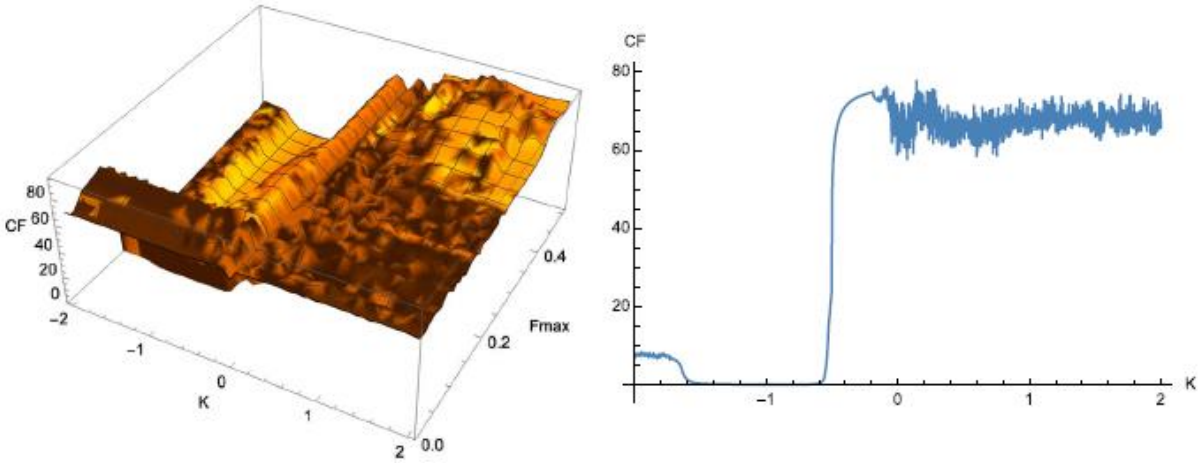

Figure 11. Examples of a CF_{simple} landscapes, stabilization of $p-1$, parametric optimization of K and F_{max} for Pyragas control method.

Indeed, it is not possible to simply perform the above-mentioned multiplication of CF_{UNI} and (NI), since in the case of full stabilization the result would be zero, regardless of the number of steps. It is, therefore, necessary to define a constant SC which, with respect to common techniques for converting a multi-criteria problem to a singlecriteria problem, will provide approximately the same numerical value (in order) for both parts of the complex cost function. The value of SC (6) is computed using $ExpCF$, which is the value of the exponent of the power of 10 of the unpenalized basis part of the cost function (5). To avoid uncertainties associated with the stabilization value returning 0, a small constant is added to the value of the cost function defining the stabilization quality. Such a design also allows the introduction of preferential weights - either for quality or speed.

$$ExpCF = \log_{10}\left(\sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| + 10^{-15}\right) \quad (5)$$

$$SC = 10^{ExpCF} \quad (6)$$

The final design of targeting CF (CF_{MULTI}) has the form (7). The metaheuristics should find the solutions to secure the fast targeting of the desired behaviour of the system.

$$CF_{MULTI} = (NI \cdot SC) + p + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| \quad (7)$$

The last possible modification is to increase the robustness for a different range of initial conditions to ensure that the found control parameters are not optimal only for the initial conditions used in the optimization. The final CF value is computed as a sum of m repeated simulations for different initial conditions, where (for example) $x_{initial}$ is from the range 0.05-0.95 and uses step 0.1 (i.e. $m = 10$). Consequently, the metaheuristics should find robust solutions securing the fast targeting of the desired behaviour of the system for almost any initial conditions. The CF used for the evolutionary approach is given in (8):

$$CF_{MULTI} = \sum_{1}^{m}\left((NI \cdot SC) + p + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t|\right), \quad (8)$$

The issue of pure searching for periodic orbits causes very chaotic, erratic, and discrete-type CF landscapes (see **Figure 14**). As a sort of compromise between the simple design of a cost function for the $p - 1$ orbit only and the above universal designs requiring knowledge of the desired state (periodic orbit), experiments were performed with functions that should allow the defined order of the periodic orbit to be found. The trade-off is more from a user's point of view since transient chaotic oscillations are still included in the calculation of the cost function. In recent publications, this type of function has been referred to as a 'black box.' Based on the choice of the target orbit, a specific cost function is selected.

In this case, it is not possible to use the simple rule of minimizing the area created by the difference between the required and actual state on the whole simulation interval. It means that this cost function design does not take any numerical target state into consideration, but the selected target behaviour of the system. The proposal is based on the following simple rule. In discrete systems, the iteration $y(n)$ and $y(n + k)$ of output value, where k is the order of unstable periodic orbit, must be the same. The idea is then to minimize the area created by the difference between the n and $n + k$ output iteration on the whole simulation interval, thus at the same time, this proposal of CF should secure fast targeting into the close neighbourhood of desired periodic orbit. Also, it is crucial to include the

penalization, which should avoid the finding of solutions, where the stabilization occurs on boundary values (here, for experiments with Henon map {-1.0,1.5}) or oscillation between them occurs. This penalization was calculated as the sum of the number of iterations, where the system output reaches the saturation boundary value. The CF_1 for stabilization on $p-1$ orbit has form (9).

$$CF_{BB1} = p_{BB} + \sum_{t=0}^{n} |diff| \tag{9}$$

where: $PBB = penalization, diff = y_{t+k} - y_t$, and k = order of desired orbit.

Table 1. Simulation results characteristics for different CF types.

CF	Advantages	Disadvantages
CF_{Simple} (3)	Wide global extreme area, simple and fast evolutionary process.	Suitable only for p-1 orbit optimization case, no constraints or penalizations could be added.
CF_{UNI} (4)	Universal CF suitable for any UPO, arbitrary constraints or penalizations could be simply added.	High nonlinearity and chaotic nature of CF.
CF_{MULTI} (7)	Universal CF suitable for any UPO with optimization of time required for full stabilization	Extremely high nonlinearity and chaotic (discrete) nature of CF.
CF_{BB} (9)	Favorable CF landscape, no required knowledge of periodic orbit position	Naive approach only for periodic orbits of k = 1–5, since p-6 already consists possible rules for p-2 and p-3

In the case of higher periodic orbits, other simple numerical criteria had to be implemented. It defines that in the case of $p-2$ orbit, there must be some (bigger) difference between the selected output iterations. Considering the fact of minimizing the CF value, the numerical criterion in (9) had to be rewritten into the suitable form (10), which is an example for $p-2$ orbit, and c - is a small constant 1.10^{-16} which was added to prevent the evolutionary optimization from crashing, since upon finding the possible solution stabilized at $p-1$ orbit it returns the division by zero.

$$\frac{1}{|y_{t+1} - y_t| + c} \tag{10}$$

The example of cost function for $p-2$ orbit CF_2 has the form (11), and $p-4$ orbit has the form (12).

$$CF_{BB2} = p_{BB} + \sum_{t=0}^{n} |diff| + \frac{1}{|y_{t+1} - y_t| + c} \tag{11}$$

$$CF_{BB4} = p_{BB} + \sum_{t=0}^{n} |diff| + \frac{1}{|y_{t+3} - y_t| + c} \tag{12}$$

The brief overview of the simulation results characteristics given as the advan-tages/disadvantages for each single cost function design is given in **Table 1**. The cost functions landscapes (and their 2D or 3D cross-sections) are depicted in **Figures 11-16**. These figures show the dependence of the cost function landscape on the selected optimized parameters of Pyragas control technique (gain constant K and limitation of perturbation F_{max}. CF_{simple} is shown in **Figures 11** and **12** (for $p-1$ and $p-2$ orbit), CF_{UNI} in **Figure 13**, CF_{MULTI} in **Figure 14**, and finally two **Figures 15** and **16** for black-box type cost functions (CF_{BB}), showing the differences between the task of finding the periodic orbit of two different types ($p-1$ and $p-2$).

4.3.Example of metaheuristic based control of discrete chaotic maps

This section contains illustrative examples of chaos control optimization results for the Henon map and Pyragas $ETDAS$ technique. **Figures 18-21** always show pairs ofbest-found results and re-runs (100 runs) for different initial conditions.

The results of the experimental simulations confirm the characteristics of the landscapes of the cost functions and their respective definitions. The optimization of the control method parameters using CF_{simple} (**Figure 17**) and CF_{UNI} (**Figure 18**) provides fast stabilization for the same initial conditions that were used for running the metaheuristics.

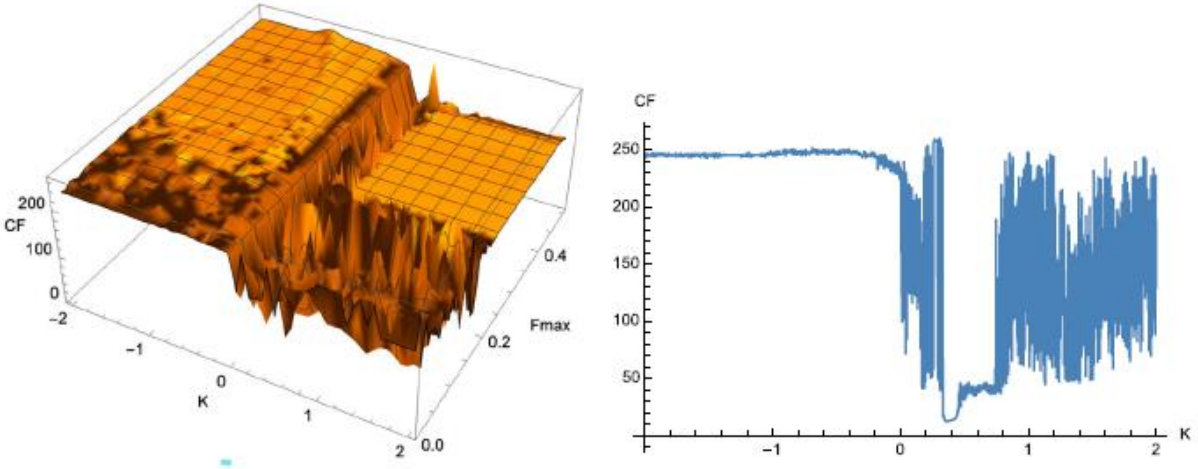

Figure 12. Examples of a CF_{simple} landscapes, stabilization of $p-2$, parametric optimization of K and F_{max} for Pyragas control method.

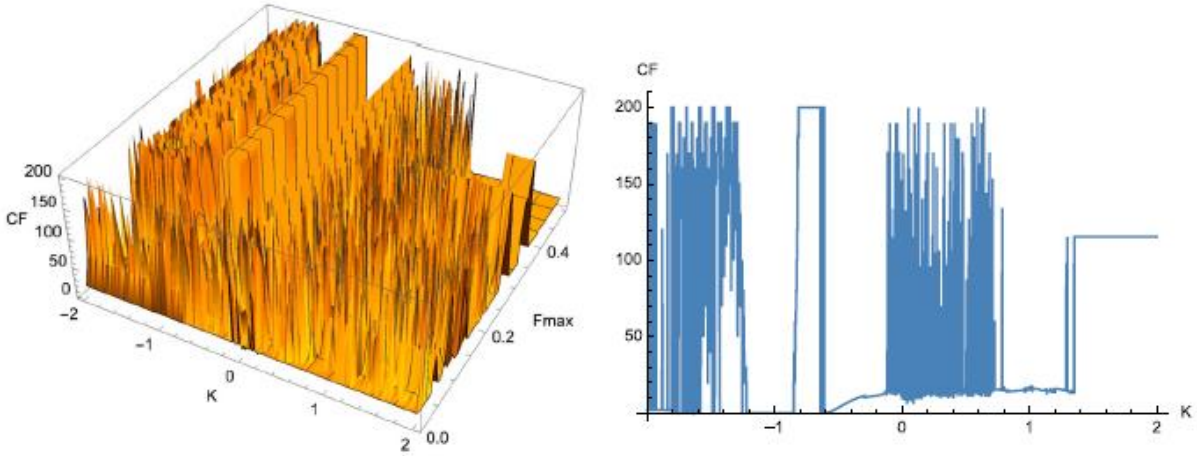

Figure 13. Examples of a CF_{UNI} landscapes, stabilization of $p-1$, parametric optimization of K and F_{max} for Pyragas control method.

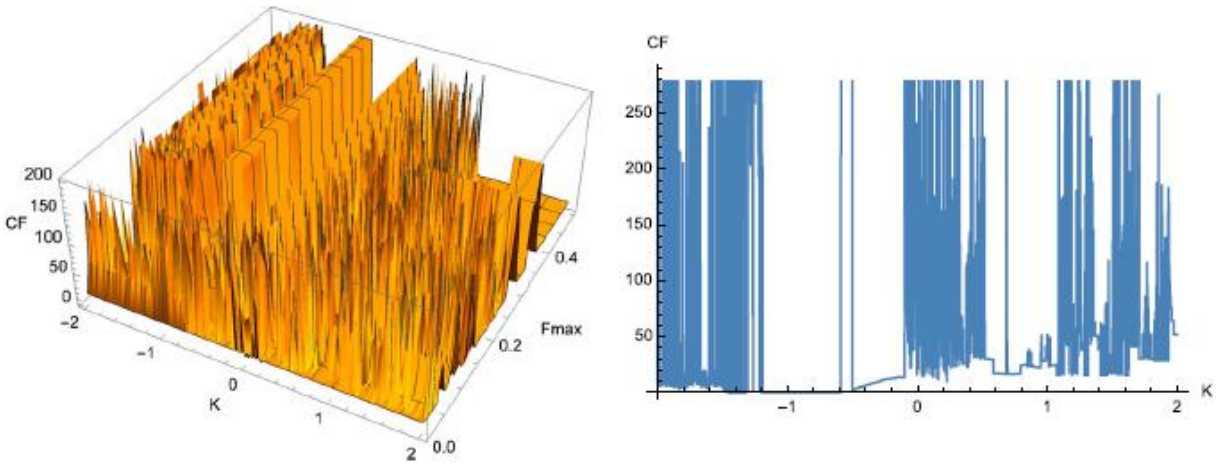

Figure 14. Examples of a CF_{MULTI} landscapes, stabilization of $p-1$, parametric optimization of K and F_{max} for Pyragas control method.

For a broader range of initial conditions, then, both approaches do not always guarantee stabilization, and in the case of CF_{UNI}, the highly chaotic landscape of the cost function and possible overfitting only to specific initial conditions (local optimum in an otherwise complex CF surface) are also manifest.

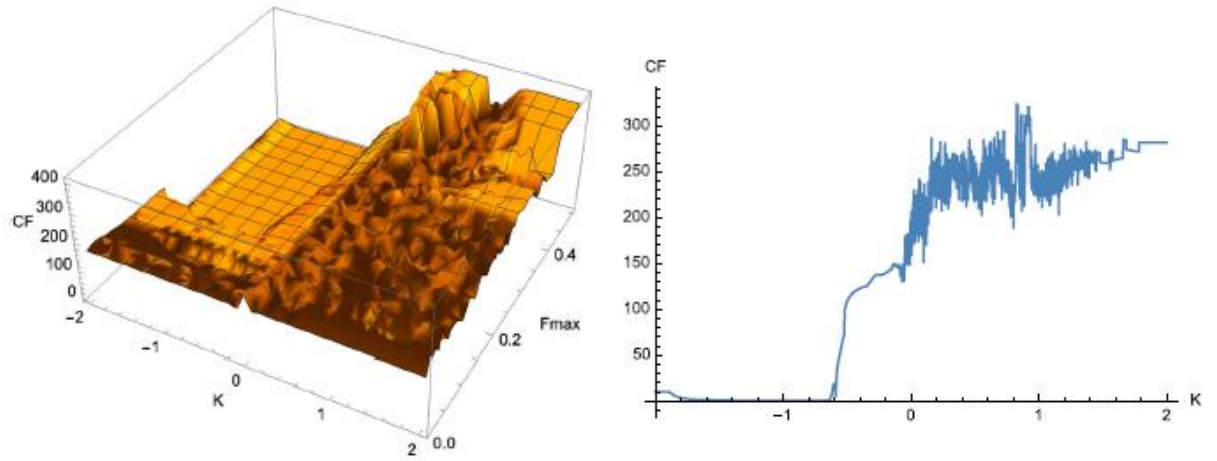

Figure 15. Examples ofa CF_{BB1} landscapes, stabilization of $p-1$, parametric optimization of K and F_{max} for Pyragas control method.

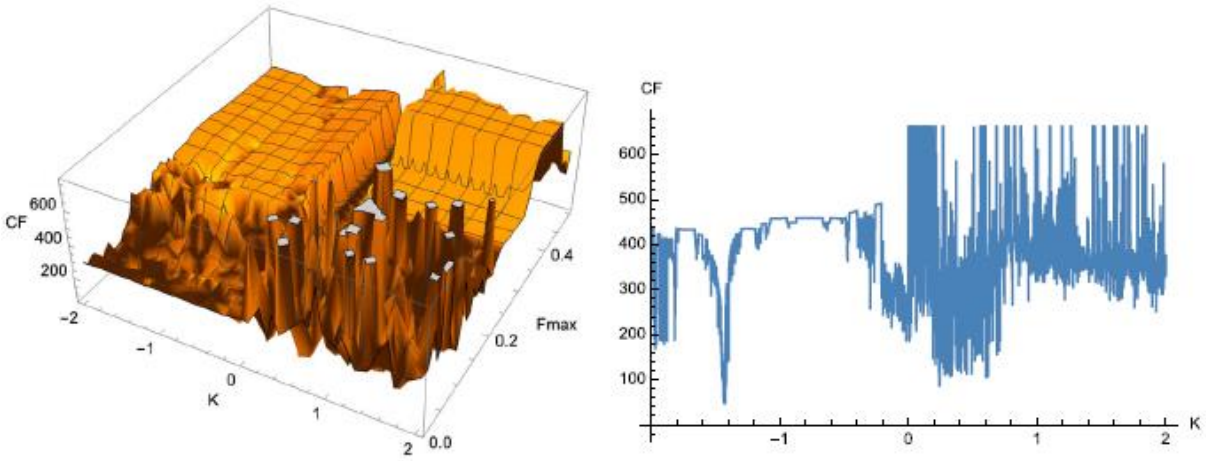

Figure 16. Examples ofa CF_{BB2} landscapes, stabilization of $p-2$, parametric optimization of K and F_{max} for Pyragas control method.

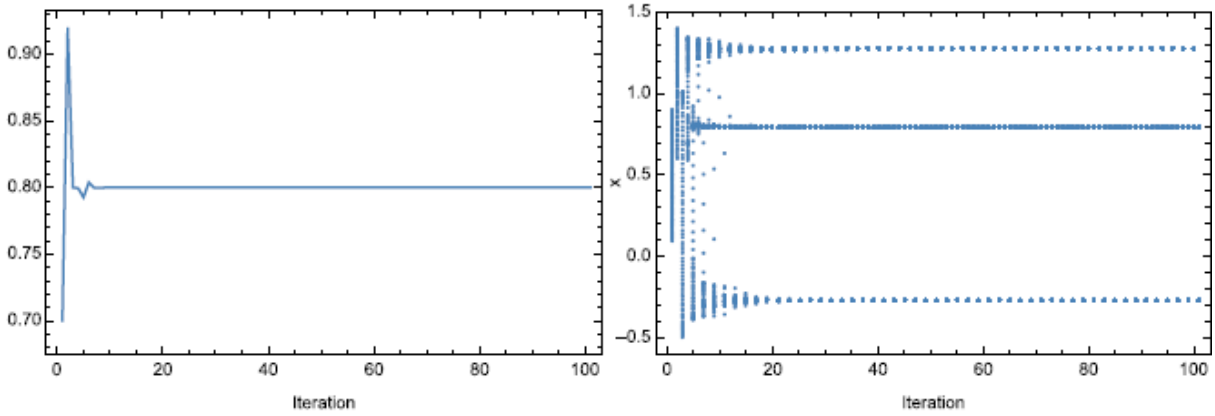

Figure 17. Stabilization of Henon map with evolutionary optimized $ETDAS$ technique, CF_{simple}.

The more computationally demanding CF_{MULTI}, then, ensures finding a robust combination of control parameters (**Figure 19**). The results of repeated simulations for different initial conditions and black box-type cost functions depicted in **Figures 20-21**, in turn, then show that simply designed to find a particular type of system behaviour may not be the best solution for this particular case, despite the favourable cost function andscape for metaheuristic optimization.

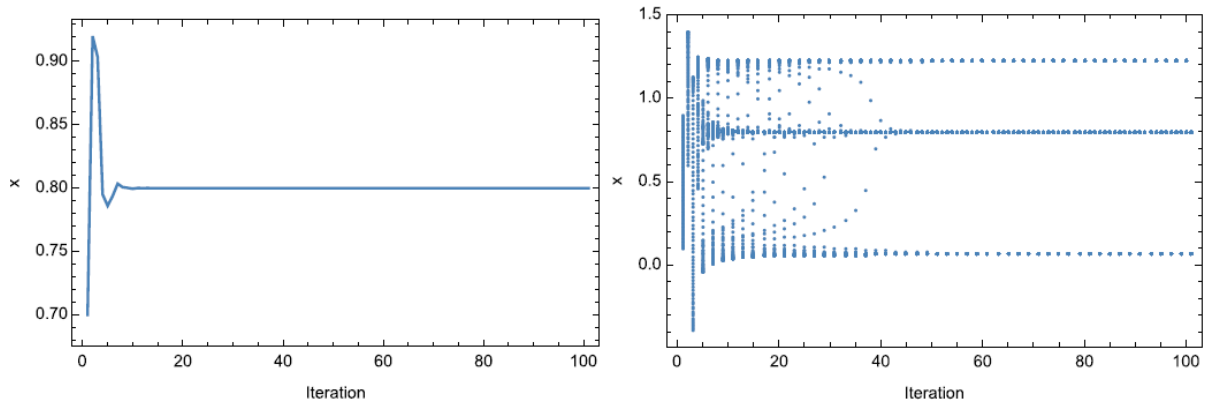

Figure 18. Stabilization of Henon map with evolutionary optimized ETDAS technique, CF_{UNI}.

In essence, the results confirm that when metaheuristics are used to control complex (nonlinear) systems, there must be a mutually appropriate fit between the definition of the cost function and the properties of its landscape.

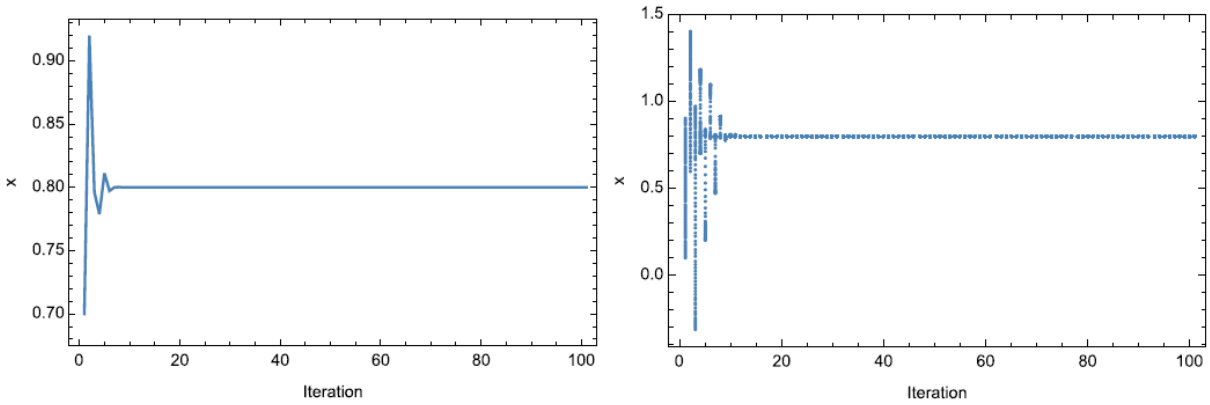

Figure 19. Stabilization of Henon map with evolutionary optimized $ETDAS$ technique, CF_{MULTI}.

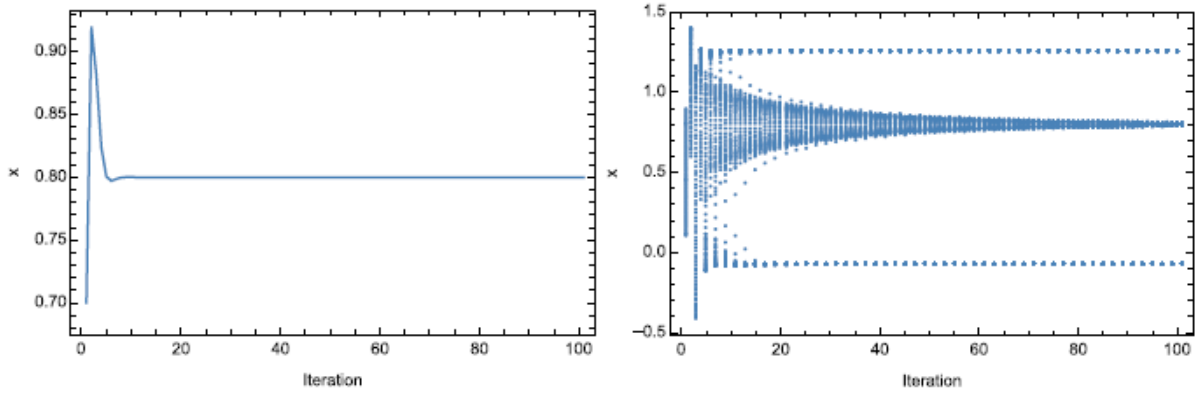

Figure 20. Stabilization of Henon map with evolutionary optimized $ETDAS$ technique, CF_{BB1}.

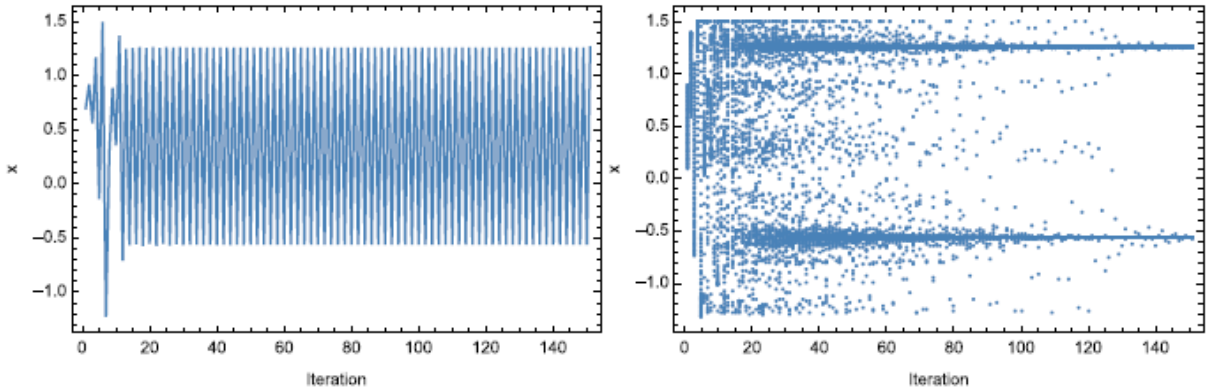

Figure 21. Stabilization of Henon map with evolutionary optimized $ETDAS$ technique, CF_{BB2}.

4.4. Control of a real chaotic system - plasma reactor control

The evolutionary algorithms were also used to control the plasma reactor in real time. Specifically, this involved active noise compensation from the signal using evolutionary techniques. This signal is generated by a Langmuir probe that was inserted in the plasma reactor. It is a diagnostic probe used to analyze the energy distribution in plasma processes. Plasmas in general, especially plasmas driven

by radio frequencies, are intrinsically nonlinear. Strong nonlinear potential waveforms around the probe embedded in the plasma, strongly distort the useful signal sensed by the probe. The cost of the research was to use evolutionary computation techniques to synthesize seven harmonic signals, with two parameters (frequency and amplitude). Thus, a total of 14 parameters have been optimized. The aforementioned seven harmonic signals were used to synthesize the resulting signal, which was intended to compensate for interference in the optimal case. The results from the experiments were published in [**36**].

4.5. Control of spatiotemporal chaos

Evolutionary computation techniques can handle more complex problems, such as even the stabilization of spatiotemporal chaos [**64**], as demonstrated in **Figures 23-25**. The cost function design was very similar to CF_{simple} (3) described in section 4.2, where the feedback pinning value has been optimized based on the difference between an actual spatiotempo-ral chaotic system and desired state. In **Figure 22** we can see an example of the landscape of the cost function, where each point of this curve represents the quality of the control intervention. The goal of the evolutionary techniques was to find the lowest point on this landscape. Of course, this is a very trivial demonstration, but the search space is still very complex. The research for example in [**45**] describes in detail the background and results in analysis.

4.6. Symbolic regression for chaos control

Another challenging task for evolutionary computation is undoubtedly symbolic regression in the feedback control rule synthesis problem. In recent publications, this approach is referred to as 'metaevolution.' The control technique (rules) can be viewed as a symbolic structure that can be synthesized according to the stabilization requirements of the chaotic system.

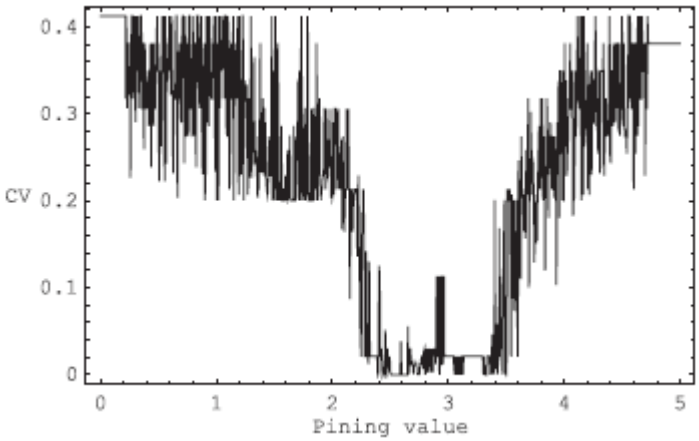

Figure 22. The CF landscape of CML chaos control by EA. The best control value (pinning) is located in the deepest position.

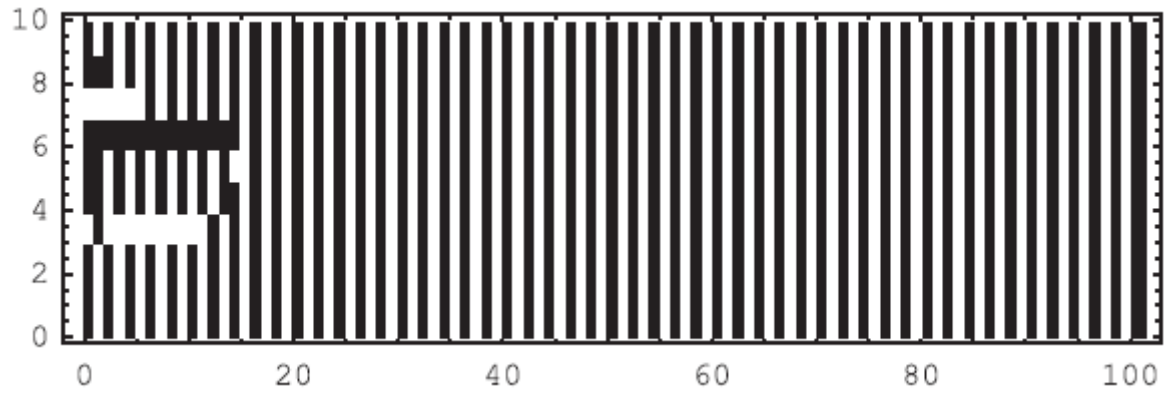

Figure 23. Stabilization and control of the spatiotemporal - CML chaotic system by EA. The x axis are iterations (time),y axis number of input sites. Two time and one spatial period stabilization are visible.

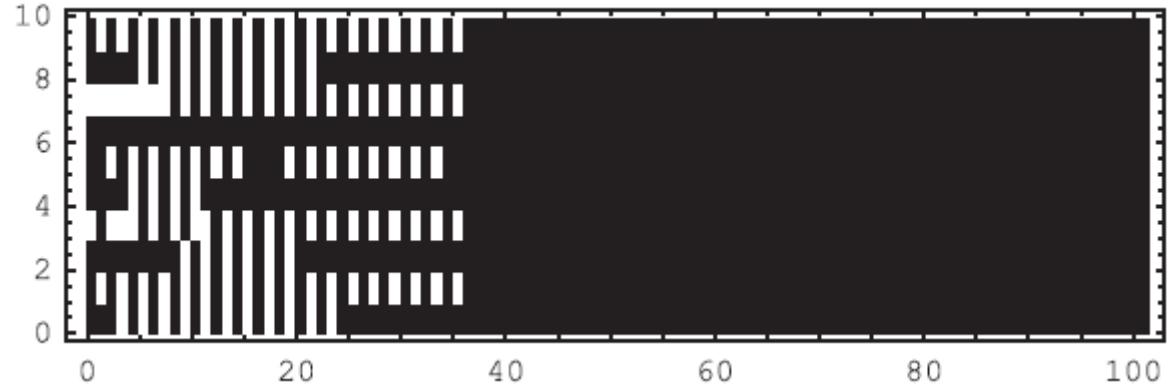

Figure 24. Stabilization and control of the spatiotemporal - CML chaotic system by EA. The x axis are iterations,y axis number of input sites. A stable state in time and space is achieved here (the black part).

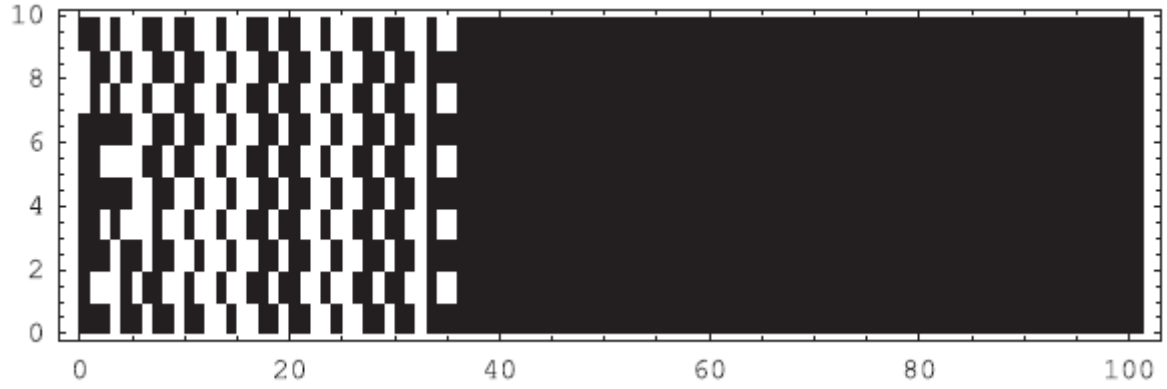

Figure 25. Stabilization and control of the spatiotemporal - CML chaotic system by EA. The x axis are iterations,y axis number of input sites. A stable state in time and space is achieved here (the black part).

The advantage is that it is not necessary to focus only on the 'common' state-of-the-art technique and estimate its optimal parameters. Using symbolic regression methods, the entire complex structure of the control method is generated, along with appropriate values of possible parameters. In addition, very specific constraints can be chosen, such as robust stabilization for selected types of chaotic systems, their regimes and initial conditions, or preferences for time, quality, and conditions for

achieving stabilization. The disadvantage is of course the increase in computational complexity. More detail on this is written in the following papers [**37, 47, 48**].

4.7. Future challenges and possibilities

Research opportunities in the field of metaheuristically based control optimization, synthesis, or synchronization of chaotic systems are certainly open for the future. Experiments with real-world problems have shown that simpler and more strongly stochastic algorithm strategies are preferable for the aforementioned tasks. The available high-performance computing capabilities then allow the use of symbolic regression optimization methods and overall more complex cost function designs. The full potential of multi-criteria optimization can also be exploited thanks to the available multi-criteria evolutionary frameworks. For the metaheuristic algorithms themselves, it is then possible to use a number of new autoconfiguration frameworks like $iRace$ [**29**] or $SMAC$ [**25**], which makes it unnecessary to laboriously search for metaheuristic hyperparameter settings.

For a more detailed description and more results regarding the interconnection between deterministic chaos and evolutionary algorithms (or any general metaheuristics), we recommend as a starting reference [**48,68**], where a large number of references to other works of similar type can be found.

5. Chaos for evolutionary computation

However, it is also necessary to focus on the evolutionary computational techniques (metaheuristics) themselves. In the last decade, it has become very popular to hybridize chaos and metaheuristics through the use of unconventional randomization schemes. Since stochastic processes are the basis of functionality in heuristic algorithms, chaotic systems are used as pseudo-random generators influencing, through their complex dynamics, e.g. the selection of individuals for the crossover process, the size of the mutation, the probabilities of performing other specific operations, and the actual movement on the hyperplane of possible solutions and the decision for the next steps. Examples of successful applications can be found [**34, 49**], benchmarking studies [**57, 71**], and exploring the impact of chaotic dynamics on population diversity or from the perspective of different types of chaotic systems [**50**].

6. Conclusion

This paper summarizes the intersection of two areas, namely deterministic chaos and so-called biologically inspired algorithms, or evolutionary computational techniques, which fall into the group of metaheuristic algorithms, and are a very important part of today's modern computer science and computational intelligence. This paper summarized two different aspects of the fusion of chaos and evolutionary computational techniques. First, the occurrence of chaos in evolutionary algorithms and swarm intelligence itself was discussed, and research opportunities are found there. In the second, the possibilities of using evolutionary computational techniques to synthesize, identify, or control chaotic systems were described, both as simulations of discrete chaotic maps and as control of real chaotic systems. Individual case studies are always briefly described, supported by graphical outputs, and possibilities for further research are discussed. In particular, in the area of evolutionary control of chaotic systems, the design of cost functions for optimization, the implications of their construction on the landscape, and the possibilities of search by evolutionary algorithms are described in detail.

Given the continuous increase in computing power, the availability of parallel computing architectures, self-configuration frameworks for the hyperparameters of the metaheuristics used, and implementations of multi-criteria algorithms, it can be argued without a doubt that this survey paper can provide a solid foundation for further significant advances and significantly better results in the effective deployment of evolutionary computational techniques in research questions and applications related to deterministic chaos.

References

[1] T. Alexeeva, Q. Diep, N. Kuznetsov, T. Mokaev, and I. Zelinka, Forecasting and control in overlapping generations model: Chaos stabilization via artificial intelligence, preprint (2022). Available at arXiv, 2208.06345.

[2] J. Alvarez-Ramirez, H. Puebla, and I. Cervantes, Stability of observer-based chaotic communications for a class oflur'e systems, Int. J. Bifurcat. Chaos 12 (**2002**), pp. 1605-1618.

[3] L. Arnold and V. Wihstutz, Lyapunov exponents: A survey, in Lyapunov Exponents: Proceedings of a Workshop held in Bremen, November 12-15, 1984, Springer, 2006, pp. 1-26.

[4] A.T. Azar and S. Vaidyanathan, Advances in Chaos Theory and Intelligent Control, Vol. 337, Springer, **2016**.

[5] T. Back and H.P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (**1993**), pp. 1-23.

[6] H.G. Beyer and H.P. Schwefel, Evolution strategies-a comprehensive introduction, Nat. Comput. 1 (**2002**), pp. 3-52.

[7] A. Byrski, E. Swiderska, J. hasisz, M. Kisiel-Dorohinicki, T. Lenaerts, D. Samson, B. Indurkhya, and A. Nowé, Socio-cognitively inspired ant colony optimization, J. Comput. Sci. 21 (**2017**), pp. 397-406.

[8] G. Chen, Controlling Chaos and Bifurcations in Engineering Systems, CRC press, **1999**.

[9] G. Chen and T. Ueta, Yet another chaotic attractor, Int. J. Bifurcat. Chaos 9 (**1999**), pp. 1465-1466.

[10] K.M. Cuomo, Synthesizing self-synchronizing chaotic systems, Int. J. Bifurcat. Chaos 3 (**1993**), pp. 1327-1337.

[11] S. Das, S.S. Mullick, and P.N. Suganthan, Recent advances in differential evolution-an updated survey, Swarm. Evol. Comput. 27 (**2016**), pp. 1-30.

[12] A. Dmitriev, A. Panas, and S. Starkov, Ring oscillating systems and their application to the synthesis of chaos generators, Int. J. Bifurcat. Chaos 6 (**1996**), pp. 851-865.

[13] A. Dmitriev, E. Efremova, L. Kuzmin, and A. Anagnostopoulos, High dimensional RC-oscillators of chaos, in Proc. of Int. Symp. NOLTA, Miyagi, Japan, Oct 28-Nov 1, 2001, pp. 139-142.

[14] M. Dorigo, V. Maniezzo, and A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 26 (**1996**), pp. 29-41.

[15] L. dos Santos Coelho and D.L. de Andrade Bernert, An improved harmony search algorithm for synchronization of discrete-time chaotic systems, Chaos Solitons Fractals 41 (**2009**), pp. 2526-2532.

[16] D. Dudkowski, S. Jafari, T. Kapitaniak, N.V. Kuznetsov, G.A. Leonov, and A. Prasad, Hidden attractors in dynamical systems, Phys. Rep. 637 (**2016**), pp. 1-50.

[17] D. Dumitrescu, B. Lazzerini, L.C. Jain, and A. Dumitrescu, Evolutionary Computation, CRC press, **2000**.

[18] R. Eberhart and J. Kennedy, A new optimizer usingparticle swarm theory, in MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39-43.

[19] K. Eguchi, T. Inoue, and A. Tsuneda, Synthesis and analysis of a digital chaos circuit generating multiple-scroll strange attractors, IEICE T. Fund. Electr. Commun. Comput. Sci. 82 (**1999**), pp. 965-972.

[20] B.I. Epureanu and E.H. Dowell, On the optimality of the ott-grebogi-yorke control scheme, Phys. D: Nonlinear Phenomena 116 (**1998**), pp. 1-7.

[21] A.E. Ezugwu, A.K. Shukla, R. Nath, A.A. Akinyelu, J.O. Agushaka, H. Chiroma, and P.K. Muhuri, Metaheuristics: A comprehensive overview and classification along with bibliometric analysis, Artif. Intel. Rev. 54 (2021), pp. 4237-4316.

[22] G. Feng and G. Chen, Adaptive control of discrete-time chaotic systems: A fuzzy control approach, Chaos Solitons Fractals 23 (**2005**), pp. 459-467.

[23] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (**1986**), pp. 533-549.

[24] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT press, **1992**.

[25] F. Hutter, H.H. Hoos, and K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in International Conference on Learning and Intelligent optimization, Springer, 2011, pp. 507-523.

[26] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Rep., Technical report-tr06, Erciyes university, engineering faculty, computer ..., 2005

[27] J.R. Koza, Genetic programming as a means for programming computers by natural selection, Stat. Comput. 4 (**1994**), pp. 87-112.

[28] N. Kuznetsov, G. Leonov, and M. Shumafov, A short survey on pyragas time-delay feedback stabilization and odd number limitation, IFAC-PapersOnLine 48 (**2015**), pp. 706-709.

[29] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, and T. Stiitzle, The iracepackage: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (**2016**), pp. 43-58.

[30] M. Lukoševičius, A practical guide to applying echo state networks, in Neural Networks: Tricks of the Trade, 2nd ed., Springer, Berlin, 2012, pp. 659-686.

[31] R. Matousek and T. Hulka, Stabilization ofhigherperiodic orbits ofthe chaotic logistic and hénon maps using meta-evolutionary approaches, in 2019IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 1758-1765.

[32] R. Matousek, L. Dobrovsky, P. Minar, and K. Mouralova, A note about robust stabilization of chaotic hénon system using grammatical evolution, in Proceedings of the 2014 International Nostradamus conference on modern methods of prediction, modeling and analysis of nonlinear systems, June 23-25, 2014, Ostrava, Czech Republic.

[33] L.Medsker and L.C. Jain, Recurrent Neural Networks: Design and Applications,CRCpress, **1999**.

[34] M. Metlicka and D. Davendra, Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems, Swarm. Evol. Comput. 25 (**2015**), pp. 15-28.

[35] S. Mirjalili, S.M. Mirjalili, and A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (**2014**), pp. 46-61.

[36] L. Nolle, I. Zelinka, A.A. Hopgood, and A. Goodyear, Comparison of an self-organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning, Adv. Eng. Softw. 36 (**2005**), pp. 645-653.

[37] Z.K. Oplatkova, R. Senkerik, I. Zelinka, and M. Pluhacek, Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic systems, Comput. Math. Appl. 66 (2013), pp. 177-189.

[38] E. Ott, C. Grebogi, and J.A. Yorke, Controlling chaos, Phys. Rev. Lett. 64 (**1990**), pp. 1196.

[39] H. Poincaré, On the problem of three bodies and equations of dynamics, Acta Math. 13 (**1890**), pp. A3-A270.

[40] K. Pyragas, Delayed feedback control of chaos, Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. 364 (**2006**), pp. 2309-2334.

[41] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, Gsa: A gravitational search algorithm, Inf. Sci. (Ny) 179 (**2009**), pp. 2232-2248.

[42] H. Richter and K.J. Reinschke, Optimization of local control of chaos by an evolutionary algorithm, Phys. D: Nonlinear Phenom. 144 (**2000**), pp. 309-334.

[43] H. Richter, An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions, in International Conference on Parallel Problem Solving from Nature, Springer, 2002, pp. 308-317.

[44] C. Ryan, J.J. Collins, and M.O. Neill, Grammatical evolution: Evolving programs for an arbitrary language, in European Conference on Genetic Programming, Springer, 1998, pp. 83-96.

[45] E. Scholl and H.G. Schuster (eds.), Handbook of Chaos Control, 2nd ed., WILEY-VCH, Weinheim, **2008**.

[46] R. Senkerik, I. Zelinka, D. Davendra, and Z. Oplatkova, Evolutionary optimisation ofhénon map control: A black box approach, Int. J. Oper. Res. 13 (**2012**), pp. 129-146.

[47] R. Senkerik, Z. Oplatkova, I. Zelinka, and D. Davendra, Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming, Math. Comput. Model. 57 (**2013**), pp. 57-67.

[48] R. Senkerik, Z. Kominkova Oplatkova, I. Zelinka, B. Chramcov, D.D. Davendra, and M. Pluhacek, Utilization of analytic programming for the evolutionary synthesis of the robust multichaotic controller for selected sets of discrete chaotic systems, Soft. Comput. 18 (**2014**), pp. 651-668.

[49] R. Senkerik, M. Pluhacek, I. Zelinka, D. Davendra, and Z. Kominkova Oplatkova, Comparison of chaos driven PSO and differential evolution on the selected PID tuning problem, in IFIP International Conference on Computer Information Systems and Industrial Management, Springer, 2015, pp. 67-76.

[50] R. Senkerik, A. Viktorin, M. Pluhacek, and T. Kadavy, On the population diversity for the chaotic differential evolution, in 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1-8.

[51] R. Storn and K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (**1997**), pp. 341-359.

[52] R. Thom, tabilité structurelle et morphogénése: essai d'une théorie générale des modéles, Mathematical physics monograph series 17 (**1972**), p. 362.

[53] T. Ueta and G. Chen, Bifurcation analysis of chen's equation, Int. J. Bifurcat. Chaos 10 (**2000**), pp. 1917-1931.

[54] T. Ushio, Synthesis of synchronized chaotic systems based on observers, Int. J. Bifurcat. Chaos 9 (**1999**), pp. 541-546.

[55] A. Vaněček and S. Čelikovský, Control Systems: From Linear Analysis to Synthesis of Chaos, Prentice Hall International (UK) Ltd., **1996**.

[56] Z. Vasicek and L. Sekanina, On area minimization of complex combinational circuits using cartesian genetic programming, in 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1-8.

[57] A. Viktorin, M. Pluhacek, and R. Senkerik, Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set, in 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 4797-4803.

[58] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, MIT press, **1999**.

[59] M.D. Vose and G.E. Liepins, Punctuated equilibria in genetic search, Complex Syst. 5 (**1991**), pp. 31-44.

[60] M.D. Vose and A.H. Wright, Simple genetic algorithms with linear fitness, Evol. Comput. 2 (**1994**), pp. 347-368.

[61] A.H. Wright and 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.

[62] S.K. Yang, C.L. Chen, and H.T. Yau, Control of chaos in lorenz system, Chaos Solitons Fractals 13 (**2002**), pp. 767-780.

[63] I. Zelinka, Investigation on evolutionary deterministic chaos control, IFAC Proc. Vol. 38 (**2005**), pp. 1101-1106.

[64] I. Zelinka, Real-time deterministic chaos control by means of selected evolutionary techniques, Eng. Appl. Artif. Intell. 22 (**2009**), pp. 283-297.

[65] I. Zelinka, Soma-self-organizing migrating algorithm, in Self-Organizing Migrating Algorithm, Springer, Cham, 2016, pp. 3-49.

[66] I. Zelinka, G. Chen, and S. Celikovsky, Chaos synthesis by means of evolutionary algorithms, Int. J. Bifurcat. Chaos 18 (**2008**), pp. 911-942.

[67] I. Zelinka, R. Senkerik, and E. Navratil, Investigation on evolutionary optimization of chaos control, Chaos Solitons Fractals 40 (**2009**), pp. 111-129.

[68] I. Zelinka, S. Celikovsky, H. Richter, and G. Chen, Evolutionary Algorithms and Chaotic Systems, Vol. 267, Springer, **2010**.

[69] I. Zelinka, D. Davendra, R. Senkerik, R. Jasek, and Z. Oplatkova,Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures, in Evolutionary Algorithms, E. Kita (ed.), InTech, 2011. ISBN: 978-953-307-171-8.

[70] I. Zelinka, M. Chadli, D. Davendra, R. Senkerik, and R. Jasek, An investigation on evolutionary reconstruction of continuous chaotic systems, Math. Comput. Model. 57 (**2013**), pp. 2-15.

[71] I. Zelinka, Q.B. Diep, V. Snášel, S. Das, G. Innocenti, A. Tesi, F. Schoen, and N.V. Kuznetsov, Impact of chaotic dynamics on the performance of metaheuristic optimization algorithms: An experimental analysis, Inf. Sci. (Ny) 587 (**2022**), pp. 692-719.

[72] T. Zhou, G. Chen, and S. Celikovsky, An algorithm for computing heteroclinic orbits and its application to chaos synthesis in the generalized lorenz system, IFAC Proc. Vol. 38 (**2005**), pp. 1079-1084.