Chapter from the book *Simulated Annealing - Single and Multiple Objective Problems*
Downloaded from: http://www.intechopen.com/books/simulated-annealing-single-and-multiple-objective-problems
1. Introduction

Simulated annealing (SA) [16], [17], [14]), belongs among those algorithms which allow steps after which the value of the objective function will deteriorate. It can thus be seen again as the local extensions of classical methods of searching. The SA is similar to hill climbing, but differs in the fact that the individual is able to overcome local extremes. However, inspiration for this formulation of statistical mechanics had been found in the description of the physical annealing process of a rigid body. In this analogy, namely during annealing of metals with unstable crystal lattice, there is a stabilization of loose particles in an optimal state, i.e. the formation of a stable crystal lattice. Such a metal has much better properties. The process is carried out by heating the metal at high temperatures to the melting point and then very slowly cooling it. Cooling is done slowly enough to eliminate unstable particles and the metal has acquired the requisite optimal quality. In the early 1980s, Kirkpatrick, Gelatt and Vecchi (Watson Research Center of IBM, USA) and independently Cerny (Department of Theoretical Physics, Comenius University in Bratislava, former Czechoslovakia) proposed solutions to the problem of finding the global minimum of combinatorial optimization analogous to the procedure of the annealing rigid body.

This chapter introduces simulated annealing in special applications focused on deterministic chaos control, synthesis and identification. The first one discusses the use of SA on evolutionary identification of bifurcations, i.e. positions of control parameters of the investigated system related to that event.

The second application discusses the possibility of using SA for the synthesis of chaotic systems. The systems synthesized here were based on the structure of well-known logistic equations. For each algorithm and its version, repeated simulations were conducted and then averaged to guarantee the reliability and robustness of the proposed method. The third and last application is focused on deterministic spatiotemporal chaos realtime control by means of selected evolutionary techniques, with SA. Realtime-like behavior is specially defined and simulated with a spatiotemporal chaos model based on mutually nonlinear joined $n$ equations, so-called Coupled Map Lattices (CML). Investigation consists of different case studies with increasing simulation complexity. For all algorithms each simulation was repeatedly evaluated in order to show and check the robustness of the methods used. All
data were processed and used in order to obtain summarized results and graphs. The most significant results are carefully selected, visualized and commented on in this chapter.

1.1. Simulated annealing

Simulated annealing ([16], and by [14]) is a generic probabilistic meta-algorithm for the global optimization problem. SA is a robust general optimization method that is based on the work of [17]. It simulates the annealing of a metal, in which the metal is heated up to a temperature near its melting point and then slowly cooled to allow the particles to move towards an optimal energy state. This results in a more uniform crystalline structure and so the process allows some control over the microstructure. SA has been demonstrated to be robust and capable of dealing with noisy and incomplete real-world data. This makes SA suitable for engineering applications.

SA is a variation of the hill-climbing algorithm. Both start off from a randomly selected point within the search space. Unlike in hill-climbing, if the fitness of a new candidate solution is less than the fitness of the current solution, the new candidate solution is not automatically rejected. Instead it becomes the current solution with a certain transition probability \( p(T) \). This transition probability depends on the change in fitness \( \Delta E \) and the temperature \( T \). Here temperature is an abstract control parameter for the algorithm rather than a real physical measure. The algorithm starts with a high temperature which is subsequently reduced slowly, usually in steps. On each step, the temperature must be held constant for an appropriate period of time (i.e., number of iterations) in order to allow the algorithm to settle into a thermal equilibrium, i.e., a balanced state. If this time is too short, the algorithm is likely to converge to a local minimum. The combination of temperature steps and cooling times is known as the annealing schedule, which is usually selected empirically. By analogy with metallurgical processes, each step of the SA algorithm replaces the actual solution by a randomly generated solution from the neighborhood, chosen with a probability depending on the difference between the corresponding function values and on a global parameter, so-called temperature \( - T \). Temperature is decreasing during the process. The current solution changes almost randomly when \( T \) is large, but increasingly downhill as \( T \) goes to zero. The allowance for uphill moves saves the method from becoming stuck at the local minimum. Simulated annealing is a stochastic algorithm depending on parameters as follow:

\[
SA = (M, x_0, N, f, T_0, T_f, \alpha, n_T),
\]

meaning of parameters is:

- **M**: space of possible solutions
- **\( x_0 \)**: initial solution, randomly selected
- **\( N(x, \sigma) \)**: Normal distribution
- **\( f \)**: The cost function
- **\( T_0 \)**: initial temperature.
- **\( T_f \)**: stopping temperature (temperature of crystallization)
- **\( n_T \)**: number of iterations of Metropolis algorithm
- **\( \alpha \)**: temperature reduction \( \alpha : T \to T', T' < T \), usually \( T' = \alpha \times T \). Parameter \( \alpha \) is usually in the range 0.8 - 0.99.
In the real world any object consists of particles. Physical state can be described by vector \( x = (x_1, x_2, ..., x_n) \) describing particle position for example. This state is related to energy \( y = f(x) \). If such system is on the same temperature \( T \) long enough, then the probability of existence of such states is given by Boltzmann distribution. The probability that the system is in state \( x \) is then given by

\[
\frac{e^{-f(x)/T}}{Q(T)}
\]

with

\[
Q(T) = \sum_x e^{-f(x)/T}
\]

where summarization is going over all states \( x \). For a sufficiently small \( T \) the probability that the system will be in state \( x_{\text{min}} \) with minimal energy \( f(x_{\text{min}}) \) is almost 1. In the 50’s simulation of annealing was suggested by means of Monte Carlo method with a new decision function 4.

\[
P(x \rightarrow x_0) = \begin{cases} 
1, & \text{for } f(x) < f(x_0) \\
e^{-\frac{(f(x)-f(x_0))}{T}}, & \text{for } f(x) \geq f(x_0)
\end{cases}
\]

The function of this decision is whether new state \( x \) (when for example one particle will change its position) is accepted or not. In the case that \( x \) is related to lower energy then the old state it is replaced by a new one. On the contrary, \( x \) is accepted with probability \( 0 < P(x \rightarrow x_0) < 1 \). If \( r \) is random number from \([0, 1]\), then new state is accepted only if \( r < P(x \rightarrow x_0) \). In (4) \( T \) has an important influence on probability \( P(x \rightarrow x_0) \) when \( f(x) \geq f(x_0) \); for big \( T \) any new state (solution) is basically accepted, for a low \( T \) states with higher energy are only rarely accepted. If this algorithm (Metropolis algorithm) is repeated for one state in a sufficient number of repetitions, then the observed distribution of generated states is basically Boltzmann distribution. This makes it possible to execute SA on a PC. The SA, repeating Metropolis algorithm for decreasing temperature then uses the final state \( T_n \) like initial state for the next iteration \( x_m \) with \( T_m = T_n - \epsilon \). Variable \( \epsilon \) is arbitrary small number. Pseudocode for SA is

Randomly selected initial solution \( x_0 \) from all possible solutions \( M \); 
\( x^* := x_0 \); 
Set initial temperature \( T_0 > 0 \); 
\( T := T_0 \); 
Select decrement function \( \alpha(t) \) and final temperature \( T_f \); 
repeat 
\( \text{for } i := 1 \text{ to } n_T \) do 
begin randomly select \( x \) from set of all possible neighbor \( N(x_0) \); 
\( \Delta f := f(x) - f(x_0) \); 
if \( \Delta f < 0 \) 
then begin \( x_0 := x \); move to a better solution is always accepted 
if \( f(x) < f(x^*) \) 
then \( x^* := x \) update the best solution 
end 
else begin randomly select \( r \) from uniform distribution on the interval \((0,1)\); 
if \( r < e^{-\Delta f / T} \) 
then \( x_0 := x \) moving to a worse solution otherwise the current solution remains unchanged
T := α(T) ;
until T < T_f ;
x* is an approximation of the optimal solution

2. Case study 1: Evolutionary identification of bifurcations

This part introduces an overview of possible use of SA on bifurcation, i.e. catastrophic events, detection (for a more detailed description, see [43]). Catastrophic events here means Thom’s catastrophes that can be used under certain conditions to model chaotic dynamics and bifurcations, that appear in the nonlinear behavior of various dynamical systems. The main aim of this work is to show that SA are capable of the bifurcations of chaotic system identification without any partial knowledge of internal structure, i.e. based only on measured data. In this part we will discuss mainly SA results. The system selected for numerical experiments here is the well-known system logistic equation derived from predator-prey system. For each algorithm and its version, simulations have been repeated 50 times. Our world, mostly consisting of nonlinear systems, is full of our i.e. human, technology that is less or more reliable. Technological systems are mostly, like their natural counterparts, nonlinear and complex and very often show chaotic as well as catastrophic behavior according to Thom’s catastrophe theory, [1], [3] and [4], that describes sudden changes in the dynamical system (well developed and described for so called gradient systems) behavior under slightly changing (usually) external conditions. These changes, depending on one or more parameters, can be modeled like the special N dimensional surfaces in the so-called parameter space, see for example [1]. As an example of such systems (and catastrophic events), we can mention systems such as electrical networks (blackout,...), economic systems (black Friday, NY stock market 1929,...), weather systems (Lorenz model of weather born via series of bifurcations modeled by Thom’s catastrophes, see [1]), civil construction failure (bridge collapse, etc), complex systems (self-criticality and spontaneous system reconstruction leading to the better energetic stability) and more. Different mathematical models, of which one possibility is the aforementioned Thom’s catastrophe theory, model such events. Our aim in this paper is to show that it is possible to use SA to identify such events on mathematical models of such systems and/or it is possible to use SA to design technological systems in such a way that the possibility to reach regimes exhibiting sudden changes in their behavior (i.e. catastrophe events) is minimized. This study is an extension of our previous research in [9] and [10].

Identification of bifurcations has been done in this research so that Lyapunov exponent is calculated in its absolute value, see Figure 6. Zero values on this cost function landscape (multiple global extremes) then indicate for what parameter A bifurcations occur. In order to locate all these zero values SA is used. They advantage is to overcome multiple local extremes that are present in used cost function. They cannot be wiped out, because of chaotic nature of studied problem.

2.1. Chaos, Thom’s catastrophes and bifurcations

Bifurcations just described in the previous section can be modeled by Thom’s catastrophe theory. This theory, [1]-[4], describes sudden changes in the system behavior under slightly changing (usually) external conditions. These changes, depending on one or more parameters, can be modeled like the special N dimensional surfaces in the so-called parameter space. Each
point in the parameter space represents one of the possible system configurations. Those points which are part of the so-called catastrophic fold (surface, plane, ...) are related to system parameter configuration when a system is changing its behavior (moment when bifurcation occurs). When a system control parameter is changed then the point in the parameter space moves and the moment when it crosses through the catastrophic fold changes, then behavior of the system is changed. This change can mean in reality changes in periodicity as well as switching to chaotic dynamic and/or also more drastic changes in the system’s physical structure and behavior. Such changes then can lead, in reality, to real catastrophes like aircrafts crashing, dam failure, collapse of a building or power network, etc. As demonstrated in [1] on the Lorenz system (weather model), born of chaos can be understood as a way through the series of bifurcations Thom’s catastrophes. Mutual relation between Thom’s catastrophes and bifurcations is thus clear.

![Catastrophe surface - Fold.](image)

**Figure 1.** Catastrophe surface - Fold.

### 2.2. Experiment setup

Experiments have been designed as described in the Table 1. Each experiment was repeated 50 times and the results are reported in the Results section. All simulations have been done on a special grid computer. This grid computer consists of 16 XServers, each 2x2 GHz Intel Xeon, 1 GB RAM, 80 GB HD i.e. 64 CPUs.

#### 2.2.1. Used algorithms

For the experiments described here (SA) [16] had been used. The control parameter settings have been found empirically and are given in Table 1 (SA).
Lyapunov exponents are another member of the family of universal features of deterministic chaos. They are numbers which basically express the divergence (or also convergence) of the state trajectories of a dynamic system. The exponents can be calculated relatively simply, both for discrete-time systems and for continuous-time systems. As will be explained later, Lyapunov exponents are closely related to the structure of the state space, which (in dynamic systems theory) is represented by an array of arrows determining the time development of the system at each point of the space. The development of the system in this space is then represented by a (usually) continuous curve [25]. Illustrative example of different behavior is depicted in Figure 3 and 4. Figure 3 shows the state space of a simple dynamic system along with two different time developments starting from two different initial conditions, which only differ by \( x = 0.01 \) in the \( x \)-axis. The behavior in the two cases is entirely different. Figure 4 shows different behavior. Hence, the behavior of a dynamical system is determined by its physical structure, which in the mathematical description is represented by the state space whose quantifiers can be Lyapunov exponents. If one is to follow colored arrows in Fig. 3, it can be noticed that they are separating with increasing time. On the other hand in Fig. 4 they after certain time occupy the same set of points in the state space, in this case called limit cycle. This observation can be described in a mathematical way by the Lyapunov exponents \( \lambda_i \), see Equation (5), see [13].

\[
\lambda_i = \lim_{t \to \infty} \frac{1}{t} \ln \frac{l_i(t)}{l_i(0)}
\] (5)

---

**Figure 2.** Catastrophe surface - Swallowtail.

| No. of particles | 100 |
|------------------|-----|
| \( \sigma \)     | 0.5 |
| \( \text{Tmin} \) | 0.000 |
| \( \text{Tmax} \) | 10000 |
| \( \alpha \)     | 0.98 |

**Table 1.** SA setting
The structure of the exponents can help assess whether chaotic behavior is present in the system or not. Lyapunov exponent is a measure of the extension or contraction of the $i_{th}$ semimajor axis of the ellipsoid. For graphic reasons, Lyapunov exponents are arranged by magnitude, i.e. $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$, where $m$ is the dimension of the phase space; this is referred to as the Lyapunov exponents spectrum. For a chaotic trajectory, one Lyapunov exponent at least must be positive, although, in addition, the existence of any asymptotic periodicity must be ruled out to confirm the chaotic nature - see, e.g., [44]. In other words, the possibility that the trajectory converges to some periodic orbit with $t \to \infty$ must be eliminated. But it is just this requirement that can pose a problem in practice if the dynamic system is investigated during a finite time interval only. Chaotic systems with more than one Lyapunov exponent are referred to as hyperchaotic, see [2]. Lyapunov exponents are closely related to Kolmogorov-Sinai entropy and kind of fractal dimensions also called Kaplan-Yorke (Lyapunov) dimension.

Figure 3. State space trajectory for a dynamic system with 2 singular points $s_1$ and $s_2$. On the position $s_1 = \{0, 0\}$ is repellor and at the position $s_2 = \{-1, 0\}$ saddle. Start points of both trajectories diverge despite fact that this coordinates ($x_1 = \{-1.56, 0.92\}$ and $x_2 = \{-1.57, 0.92\}$) are very close.

Figure 4. Different behavior can be observed when both trajectories will start in different part of state space. Despite its bigger difference in starting possition ($x_1 = \{0.4, 0.4\}$ and $x_2 = \{0.8, 0.4\}$) trajectories merge together after certain time.

2.2.3. The cost function

The main idea of this part is to identify bifurcation (in the general sense related to Thom’s catastrophes, see [1]) by means of EAs. For this purpose a logistic equation has been selected (simplified model of predator-prey system), see (6) and [5]. Logistic equation is very simple model of predator-prey system, that exhibit rich set of complex behavior for different values of control parameter $A$. For more details see [5].

$$x_{n+1} = Ax_n (1 - x_n)$$ (6)
The bifurcation diagram, [5] (i.e. system dependence on control parameter), of this system is depicted in Figure 7. The Lyapunov exponents of that system, related to parameter $A$, are depicted in 5 and 6. Simply: when the Lyapunov exponent is negative, the system has deterministic behaviour, when it is positive, the system is chaotic. When the Lyapunov exponent $\lambda = 0$ then bifurcation can be observed in the system behaviour. Then, to find these moments, we need to locate those parameters $A$ for which the Lyapunov exponent $\lambda = 0$. It is enough to use absolute value and then we get Figure 5 and 6. Figure 6 represents the cost function landscape, where values with Lyapunov exponent $= 0$ are related settings of $A$ for which bifurcation occurs. One can see that this surface is very erratic chaotic, and thus suitable candidates to find cost values with $0$ are heuristics such as evolutionary algorithms.

Cost function, used for depicting Figure 6, is given by (8) which is an absolute value of (7), see [5], variable $d$ in (7) is the difference between two nearby trajectories. Function $f(x)$ in Equation 7 is in this case logistic equation 6. Graphically is Equation 7 and 8 depicted in Figure 5 and 6. EAs were used to search for $f_{\text{cost}} = 0$. Difference between Equation 5 and 7 is mainly in fact that Equation 5 calculate $\lambda$ from function describing given system, while Equation 7 is based on recorded numerical data.

\[
\lambda = \frac{1}{n} \ln \frac{d_n}{d_0} \\
\quad d_0 = |x_i - x_j| \\
\quad ... \\
\quad d_n = |x_{i+n} - x_{j+n}|
\]

\[f_{\text{cost}} = |\lambda| \tag{8}\]

![Ljapunov exponent for Logistic equation](image)

**Figure 5.** Dependence of Lyapunov exponent on parameter $A$ of Equation (6), see (7)

2.2.4. Experimental results

Results obtained in all experiments are reported in Table 2 (SA) and Table 3 that show minimal, maximal and average cost values given by experimentation. Table 4 shows localized positions of bifurcations.
Figure 6. An absolute value of dependence of Lyapunov exponent on parameter A Equation (6), see (8)

Table 2. SA Cost Function Evaluations

| Min     | Average | Max      |
|---------|---------|----------|
| $2 \times 10^{-9}$ | $9.2 \times 10^{-6}$ | $2 \times 10^{-2}$ |

Table 3. Cost values

| Value of parameter $A$ | 3.543 | 3.568 | 3.632 | 3.742 | 3.847 |
|------------------------|-------|-------|-------|-------|-------|
| Hit                    | 23    | 12    | 32    | 19    | 14    |

Table 4. The coordinates (value of parameter $A$) of bifurcations (upper row) and number of its successful (approximative) identifications (bottom row).

Figure 7. Composed picture show identified bifurcations (red dashed lines) that are in exact correlation with Lyapunov exponent, compare Figure 5 and 6 and Table 4, also [9], [10].
SA in basic versions has been used in a simple system (6), called logistic equation, to localize bifurcations. Simulations were repeated 50 times. Based on results reported in the previous section it can be stated that:

1. **Simulations.** All simulations has been successfully finished by localization of the bifurcation event, see Table 4.

2. **Precision.** Located bifurcations were localized with high precision, however, not exactly, see Figure 7-10. Geometrically it can be interpreted so that the system is at the edge of a catastrophic model or almost before crossing the fold on the surface of Thom’s catastrophe manifold. Our opinion is that such non-precise localization can be useful, when applied to a real system and bifurcation can cause real damage, which means that the evolutionary search for bifurcations on a real system can be stopped very near to the bifurcation position. Figures 7-10 clearly show, that there are a lot of other bifurcations that were not localized by the SA that was used, which imply better settings of SA or acceptable value of $\lambda$ (see Localized bifurcations below), is probably possible. This is open question for further experiments.
3. **Chaotic systems.** Besides well known logistic equation, other systems such as Ikeda, Burger, Delayed Logistic, etc. can be selected. The logistic equation system (derived from predator prey system) has been chosen because it is well known, and well investigated. Other systems (like those mentioned above) are now in the process of investigation.

4. **Simulation environment and software.** Software used for all calculations, numerical simulations and visualizations was the well known Mathematica. Thanks to fact that Mathematica is an integrated environment, the time scale for each simulation was in minutes. It is clear that when C++ or another fast programming language would be used, then time scale of simulation should be much shorter and thus real use of this approach on real systems is possible.

5. **Localized bifurcations.** It is important to note that only main bifurcations have been localized. Bifurcations at exact positions were undiscovered, which implies space for future research, i.e. better algorithm setting, cost function improvement, etc. Figures 7-10 clearly show, that there are a lot of other bifurcations that were not localized by the SA, so the question is: what is the limit for EAs to be used on this kind of task, or what will be better settings for it? The bifurcations were determined by the same SA in repeated simulations. Localized bifurcation implies that $\lambda = 0$, however because SA is numerical algorithm, working with certain preciseness, successful localization has been set for $\lambda < 10^{-1}$.

In the light of our previous experiments described in [11], [12] and [13] it is clear that evolutionary techniques are capable of solving a wider class of problems in the chaos research domain.

3. **Case study 2: Chaos synthesis**

This part introduces the notion of chaos synthesis by means of evolutionary algorithms and develops a new method for chaotic systems synthesis, [13]. This method is similar to genetic programming and grammatical evolution and is being applied along with evolutionary algorithms: differential evolution, self-organizing migrating, genetic algorithm, simulated annealing and evolutionary strategies. The aim of this investigation is to synthesize new
and simple chaotic systems based on some elements contained in a pre-selected existing chaotic system and a properly defined cost function. The investigation consists of two case studies based on the aforementioned evolutionary algorithms in various versions. For all algorithms, 100 simulations of chaos synthesis were repeated and then averaged to guarantee the reliability and robustness of the proposed method. The most significant results were carefully selected, visualized and commented on in this chapter.

Deterministic chaos, discovered by [21] is a fairly active area of research in the last few decades. The Lorenz system produces a well-known chaotic attractor in a simple three-dimensional autonomous system of ordinary differential equations, [21], [22]. For discrete chaos, there is another famous chaotic system, called the logistic equation [23], which was found based on a predator-prey model showing complex dynamical behaviors. These simple models are widely used in the study of chaos today, while other similar models exist (e.g., canonical logistic equation [24] and 1D or 2D coupled map lattices, [25]). To date, a large set of nonlinear systems that can produce chaotic behaviors have been observed and analyzed. Chaotic systems thus have become a vitally important part of science and engineering at the theoretical as well as practical levels of research. The most interesting and applicable notions are, for example, chaos control and chaos synchronization related to secure communications, amongst others.

The aim of the present investigation is to show that EA-based symbolic regression (i.e., handling with symbolic objects to create more complex structures) is capable of synthesizing chaotic behavior in the sense that mathematical descriptions of chaotic systems are synthesized symbolically by means of evolutionary algorithms. The ability of EAs to successfully solve this kind of black-box problems has been proven many times before (see, for example, [26], [27]), and is proven once again here in this paper.

3.1. Cost function

The cost function was in fact a little bit complex decision function. The cost function used for chaos synthesis, comparing with other problems is quite a complex structure which cannot be easily described by a few simple mathematical equations. Instead, it is described by the following algorithm procedure:

1. Take a synthesized function and evaluate it for 500 iterations with and a sampling step of $\Delta A = 0.1$.
2. Check if each value of $A$ for all 500 iterations is unique or if some data are repeated in the series (the first check for chaos, indirectly). If the data are not unique, then go to step 5 else go to step 3.
3. Take the last 200 values, and for each value of $A$, calculate its Lyapunov exponent.
4. Check the Lyapunov exponent: If the Lyapunov exponent is positive, write all important data (synthesized functions, number of cost function evaluations, etc.) into a file. Then, repeat the simulation for another synthesized system by going to step 1.
5. If the data are not unique, i.e., if the Lyapunov exponent is not positive, return an individual fitness, and sum all values whose occurrences in the dataset from step 1 are more than 1 (simply, it returns the occurrences of periodicity, quasi-periodicity – higher penalization of an individual in the evolution).
More brief and simple description of above algorithmically defined cost function can be also done as in Equation 9.

\[
\text{Data}[f_{\text{syt}1}, \ldots, f_{\text{syt}500}] := f_{\text{syt}k+1} = f_{\text{syt}k}(x_{\text{start}}), \quad k \in [1, 500]
\]

\[
\begin{cases}
\text{if } \text{Data}[f_{\text{syt}1}] \neq \text{Data}[f_{\text{syt}2}] \neq \ldots \neq \text{Data}[f_{\text{syt}500}] \\
\text{then } \text{calculate } \lambda \text{ for } \text{Data}[f_{\text{syt}300}, \ldots, f_{\text{syt}500}], \text{ if } \lambda > 0 \text{ write all to file} \\
\text{else penalize individual}
\end{cases}
\]

The input to this cost function is a synthesized function and the output is the fitness (quality) of the synthesized function (i.e., the individual in the population). In the cost function, it was tested twice to see if the behavior of the just-synthesized formula is really chaotic. The first test was done in step 2 (unique appearance in the data series) and the second one, in step 4, where the Lyapunov exponent was tested numerically [5].

The reason why in step 5 the sum of the non-unique data appearances was returned is based on the fact that the evolution is searching for minimal values. In this case, the value 2 means that some data element appears in the 500-data series twice, and 1 would means that there is no periodicity and thus synthesized system is possible candidate for chaos.

To make sure that the results so obtained are correct, all written synthesized functions were used for automatic generation of bifurcation diagrams and Lyapunov exponents, as further discussed below.

### 3.2. Simulations and results

SA algorithms have been applied 100 times in order to find artificially synthesized functions that can produce chaos. All of these experiments were done using the Mathematica software. The primary aim of this comparative study is not to show which algorithm is better or worse, but to show that symbolic regression is able to synthesize some new (at least in the sense of mathematical description and behavior) chaotic systems.

Based on the results from experiments, two different sets of figures were created. The first set shows the performances of different algorithms from different points of view (see [13]), the second set (Figure 11 - Figure 14) shows behaviors of the selected synthesized programs, i.e., bifurcation diagrams. The synthesized programs are also appended to each figure in the form of the mathematical formula. Figure 15 shows an example of the so-called program length histogram, generated from 100 simulations. Program length (in Mathematica command: LeafCount, denoted as LC) means a number of elements that create a mathematical formula.

As a summary, the following statements are presented:

As a summary, the following statements are presented:

- **Result verification.** To be sure that the results as presented in the paper are correct, all written synthesized functions were used for automatic generation of bifurcation diagrams and Lyapunov exponents.

- **Simulation results.** Based on the results (see selected Figures 11 - 14) and the selected bifurcation diagrams, it can be stated that all simulations give satisfactory results and that evolutionary synthesis of chaos is capable of solving this class of problems.

- **Range of chaos and interval of observation.** During evolutions, chaos was searched by focusing on interval [0, 4], based on the a priori known behavior of the logistic equation.
whose elements were used in the evolution. Despite the a priori known information, a few chaotic systems were located also outside of this interval. That was due to the fact that a part of the chaotic behavior was inside the interval [0, 4] and thus EA was able to identify it. From these facts, it is clear that EA are able to locate chaos in a wider range than those expected from some textbook exemplary systems.
Exemplary system synthesis. Based on the fact that the logistic equation (its elements and range) is used for chaos synthesis, it is logical to expect that during evolution (if repeated many times) the original system should also be synthesized. That event was also observed a few times, exactly in the mathematical form Equation (6). Some selected bifurcation diagrams of synthesized systems are depicted in Figures 16 - 19.

Mutual comparison. When comparing all algorithms, it is obvious that these algorithms produced good results. Parameter setting for the algorithms was based on a heuristic approach and thus there is a possibility that better settings can be found there. Based on these results, it is clear that for symbolic synthesis via analytic programming any evolutionary algorithm can be used.

Engineering design. It is quite clear that evolutionary synthesis of chaos can be applied to engineering design of devices based on chaos (signal transmission via chaos, chaos-based encryption, and so on). Based on principles and results reported in this paper, it should be possible to synthesize systems with some precisely defined chaotic features and attributes.
4. Case study 3: Evolutionary control of CML systems

This contribution introduces a continuation of an investigation on deterministic spatiotemporal chaos realtime control by means of selected evolutionary techniques. Realtime-like behavior is specially defined and simulated with a spatiotemporal chaos model based on mutually nonlinear joined $n$ equations, so called Coupled Map Lattices (CML), see [25]. SA algorithms have been used for chaos control here. For modeling of spatiotemporal chaos behavior, so-called coupled map lattices were used based on a logistic equation to generate chaos. The main aim of this investigation was to show that evolutionary algorithms, under certain conditions, are capable of control of CML deterministic chaos, when the cost function is properly defined as well as parameters of a selected evolutionary algorithm. Investigation consists of four different case studies with increasing simulation complexity.
For all algorithms each simulation was repeated 100 times to show and check the robustness of the methods used. All data were processed and used in order to obtain summarizing results and graphs. Many methods were adapted for the so-called spatiotemporal chaos represented by coupled map lattices (CML). Control laws derived for CML are usually based on existing system structures, Schuster H. G. (1999), or using an external observer, Chen G. (2000). Evolutionary approach for control was also successfully developed in, for example, Richter H. and Reinschke K. J. (2000), Richter H. and Reinschke Kurt J. (2000a), Richter H. (2002), Zelinka I. (2006). Many published methods of deterministic chaos control (DCC) were (originally developed for classic DCC) adapted for so-called spatiotemporal chaos represented by CML, given by (10). Models of this kind are based on a set of spatiotemporal (for 1D, Figure 20) or spatial cells which represents the appropriate state of system elements. A typical example is CML based on the so-called logistic equation, [23], [5], [29], which is used to simulate the behavior of system which consists of n mutually joined cells via nonlinear coupling, usually noted as ε. A mathematical description of the CML system is given by Equation (10). The function which is represented by \( f(x_n(i)) \) is arbitrary discrete system in this case study logistic equations have been selected to substitute \( f(x_n(i)) \).

\[
x_{n+1}(i) = (1 - \epsilon)f(x_n(i)) + \frac{\epsilon}{2}(f(x_n(i - 1)) + f(x_n(i + 1)))
\]

The main aim of this part is to show that evolutionary algorithms (EA) are capable of controlling (as was also shown for temporal DCC in [31], [32] CML as well as deterministic methods without internal system knowledge operating with CML as with a black box. The
ability of EAs to successfully work with a problematic kind of black box have been proven; see for example realtime control of plasma reactor, [34], [35], [36] or CML non realtime control by evolutionary algorithms [37], [38], [39]. This part is organized as follows. The first part outlines the motivation of this investigation.

This is followed by a brief survey of evolutionary algorithms which follow, along with a brief description of the idea of CML chaos control and the evolutionary algorithms that were used. Evolutionary chaos control is then studied, and finally experimental results are reported, followed by the conclusion.

The main question in the case of this participation was whether EAs are able to control and stabilize chaotic systems like CML, and if they are able to control CML like a black box system, i.e. when the structure of controlled system is unknown. All experiments here were designed to check this idea and confirm or refute this idea. Comparison has been done with a control based on analysis of a CML system, [40], [28] and analytic derivation of control law for CML. Behavior of a controlled CML is as demonstrated in Figure 21 - Figure 23. A snapshot of front-wave stabilization of a CML is depicted here. Figure 21 is the initial phase of front-wave. It is clearly visible that it is fully random. Figure 21 shows the CML after 60 iterations a pattern-like structure is visible there. The last snapshot was made after 344 iterations the CML has been successfully stabilized. Thus, the main aim was to stabilize CML with the quality as standard controlling techniques.

![Figure 21. Successfull stabilization of CML in T1S3 pattern - start.](image)

### 4.1. CML control

#### 4.1.1. Parameter setting

The control parameter settings have been found empirically and are given in Table 5. The main criterion for this setting was to keep the same setting of parameters as much as possible and, of course, the same number of cost function evaluations as well as population size. Individual length represents the number of optimized parameters (number of pinning sites, values...). Compared to previous [38], the length of experiments has been an individual set of 1 or 2, according to the case study. In [38] and [41], individual length was equal to the number of pinning sites, which has increased complexity of calculations. To simplify simulations here, a simple presumption has been taken into consideration instead of an exact number of pinning sites as in [38], their periodicity has been estimated in the evolution, i.e. if parameter for
Figure 22. Successful stabilization of CML in T1S3 pattern - iteration 60.

Figure 23. Successful stabilization of CML in T1S3 pattern - pattern is stabilized.

the pinning site was, for example, 4th, then each 4th site has been used for pinning value application, etc... In total 4 case studies has been selected and used in this experiment:

- **Case Study A**: Pinning Value Estimation.
- **Case Study B**: Pinning Sites Position Estimation.
- **Case Study C**: Minimal Pinning Sites Position Estimation.
- **Case Study D**: Minimal Pinning Values and Sites Position Estimation.

see [13].

4.2. Used hardware

CML control in this case study has been done on a special grid computer, as opposed to a simple PC as in [41]. This grid computer, called Emanuel, consists of two special Apple servers, the bigger one is based on 16 X Servers, each 2x2 GHz Intel Xeon, 1 GB RAM, 80 GB HD i.e. 64 CPUs. The second one is created from 7 Apple Minimacs CoreDuo i.e. number of accessible CPUs is 14. In total there were 78 CPUs available. Emanuel was used for calculations in two ways. The first one was focused on use of each CPU like a single processor and thus a rich set of statistically repeated experiments were not a time problem. In the second
way Emanuel was used like grid machine to increase speed of selected simulations reported in this book. This does not mean that this class of problems can be solved only on special computers. All solved problems and reported case studies in this book can also be done on a single PC. Of course in different time scale.

4.2.1. Cost function

The fitness (cost function) has been calculated by using the distance between the desired CML state and actual CML output, Equation (11). The minimal value of this cost function, which guarantees the best solution, is 0. The aim of all simulations was to find the best solution, i.e. a solution that returns the cost value 0. This cost function was used for the first two case studies (pinning values setting, pinning sites setting). In the next (last) two case studies, cost function (12) was used. It is synthesized from cost function (11) so that two penalty terms are added. The first one, $p_1$, represents the number of pinning sites used in the CML. The second one, $p_2$, is added here to attract the attention of the evolutionary process on the main part of cost function. If this were not done, then mainly $p_1$ would be optimized and the results would not be acceptable (proved by simulations). Indexes $i$ and $j$ are coordinates of the lattice element, $CML_{i,j}$ is $i_{th}$ site (equation) in $j_{th}$ iteration. For all simulations of $T_1S_1$ the stabilized state was set to $S_1 = 0.75$, and for $T_1S_2$ to period $S_2 = (0.880129, 0.536537)$, i.e. CML behavior was controlled to this state.

Knowledge (at least approximate) about complexity and variability of the cost function used is very important. Such knowledge can be important when the class of optimizing algorithms is selected. Thus a few ideas and examples have been selected here to show complexity and its dependence on chaotic system parameter setting. How complex such a cost function can be is clearly visible from Figure 24 and Figure 25. It is clearly visible, that cost function is partly chaotic and for a certain pinning value, global minimum representing stabilization is accessible. The chaoticness of such a graphical representation is caused by the fact that calculations are based on a chaotic system. If the average value (over many of such runs) were to be calculated, then we would get graphs with a smooth curve. However, because our simulations were running on a single run, not over a lot of them, Figure 24 and Figure 25 are real representations of the landscape of our cost function. Our simulations were run over such, or a similar, landscape. It is also important to note that for each simulation of CML, the exact shape and its chaoticness can be slightly different from the previous one, due to the sensitivity of initial conditions. It is clear that the complexity of the cost function that was used is big, despite the simple mathematical description. Also, suitable stabilizing combination of control parameters depends on the number of CML iterations (after a certain number of iterations the

| Case Study | A | B | C | D |
|------------|---|---|---|---|
| No. of particles | 10 | 10 | 10 | 10 |
| $\sigma$ | 0.5 | 0.5 | 0.5 | 0.5 |
| $k_{\text{max}}$ | 66 | 66 | 66 | 66 |
| $T_{\text{min}}$ | 0.0001 | 0.0001 | 0.0001 | 0.0001 |
| $T_{\text{max}}$ | 1000 | 1000 | 1000 | 1000 |
| $\alpha$ | 0.95 | 0.95 | 0.95 | 0.95 |
| Individual Length | 1 | 1 | 1 | 2 |

Table 5. SA setting for case studies A, B, C and D
combination is permanent) and configuration \((T_1S_{1,2})\) of stabilized state.

\[
f_{\cos t} = \sum_{i=1}^{30} \sum_{j=a}^{b} |TS_{i,j} - CML_{i,j}|^2
\]

\(TS_{i,j}\) – target state of CML

\(CML_{i,j}\) – actual state of controlled CML

\(\{a, b\}\) = \(\{80, 100\}\) for \(T_1S_1\) and \(\{a, b\}\) = \(\{580, 600\}\) for \(T_1S_2\)

\[
f_{\cos t} = p_1 + \left( p_2 + \sum_{i=1}^{30} \sum_{j=a}^{b} |TS_{i,j} - CML_{i,j}| \right)^2
\]

\(TS_{i,j}\) – target state of CML

\(CML_{i,j}\) – actual state of controlled CML

\(p_1\) – number of actually selected pinning sites

\(p_2\) – 100, heuristically set weight constant

\(\{a, b\}\) = \(\{80, 100\}\) for \(T_1S_1\) and \(\{a, b\}\) = \(\{580, 600\}\) for \(T_1S_2\)

Figure 24. Successful stabilization of CML (30 × 100 - 30 pinning sites, 100 iterations) in \(T_1S_1\) pattern - stabilization after 52 iterations is visible.

Figure 25. CML \(T_1S_2\) in configuration 30 × 600 – stabilization after 400 iterations is visible.

4.3. Experimental results

All simulations were done in the framework of four case studies A, B, C and D. In all numerical case studies both CML regimes \(T_1S_1\) and \(T_1S_2\) were compared to show how the
complexity of both regimes can influence the number of cost function evaluations etc. The method of evolutionary deterministic chaos control described here is relatively simple, easy to implement and easy to use. Based on its principles and its possible universality it can be stated that evolutionary deterministic chaos control is capable of solving this class of CML deterministic chaos control problems. The main aim of this part was to show how various CML control problems were solved by means of evolutionary algorithms. Evolutionary deterministic chaos control was used here in four basic comparative simulations. Each comparative simulation was repeated 100 times and all results (see [13]) were used to create graphs for performance evaluation of evolutionary deterministic chaos control. They were chosen to show that evolutionary deterministic chaos control can be regarded as a blackbox method and that it can be implemented using arbitrary evolutionary algorithms.

5. Conclusions

In this chapter selected applications of simulated annealing were briefly mentioned, such as: evolutionary identification of bifurcations, synthesis of chaotic systems and evolutionary control of spatiotemporal CML systems. All experiments mentioned here are a part of a more extended set of experiments, so we recommend that interested readers read our original sources for a full description. It has been numerically demonstrated that SA is still a useful algorithm, capable of solving various problems from engineering as well as from theory.

Acknowledgement

This work was supported by the Development of human resources in research and development of latest soft computing methods and their application in practice project, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

Author details

Ivan Zelinka and Lenka Skanderova
Department of Computer Science, Faculty of computer science, 17. listopadu 15, 708 33 Ostrava-Poruba, VSB-TUO Ostrava, Czech Republic

6. References

[1] Gilmore R., Catastrophe Theory for Scientists and Engineers, John Wiley and Sons, 1993
[2] Rössler O (1979) An equation for hyperchaos. Phys. Lett. A, 71:155
[3] Arnold V. (1991). The Theory of Singularities and Its Applications, Accademia Nazionale Dei Lincei, Pisa, Italy, 1991
[4] Poston T, Stewart I. (1977). Catastrophe Theory and its Applications, 842-844, IEEE Press, New York, USA, 1977
[5] Hilborn R. (1994). Chaos and Nonlinear Dynamics, Oxford University Press, 1994
[6] Price K., Storn R. et al. (1994). Differential Evolution - A Practical Approach to Global Optimization, Springer-Verlag, 2005
[7] Zelinka I. (2004). SOMA - Self Organizing Migrating Algorithm, In: New Optimization Techniques in Engineering, Onwubolu, Babu B., (Ed.), 167-218, Springer-Verlag, New York, 2004
[8] Davis L. (1996). Handbook of Genetic Algorithms, Van Nostrand Reinhold, Berlin, 1996

[9] Zelinka I., Davendra D., Senkerik R., Jasek R., Oplatkova Z. (1996). Evolutionary Techniques And Its Possibility To Identify Catastrophic Events, In Proceeding of the 5th Global Conference on Power and Optimization, 983-44483-49, Dubai, UAE, June 1-3 2011

[10] Zelinka I., Davendra D., Senkerik R., Jasek R.(2011). Evolutionary Techniques And Its Possibility To Identify Catastrophic Events - An Extended Study, Matousek Radek: Mendel 2011 -17th International Conference on Soft Computing, pp. 73-79, ISBN 978-80-214-4302-0, Brno University of Technology, 2011a

[11] Ivan Zelinka, Roman Senkerik, Eduard Navratil (2006). Investigation On Realtime Deterministic Chaos Control By Means Of Evolutionary Algorithms, , Proceeding of the Chaos 06, June 22-24 2006, Reims, France

[12] Ivan Zelinka, Roman Senkerik, Eduard Navratil (2009). Evolutionary Identification of Chaotic System, Proceeding of the Chaos 09, June 22-24 2009, London.

[13] Zelinka I, Celikovsky S, Richter H and Chen G., (2010). Evolutionary Algorithms and Chaotic Systems, pp.550, Springer, Germany, 2010

[14] Cerný V., Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, J. Opt. Theory Appl., 45, 1, 41-51, 1985

[15] Goldberg, David E.(1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company Inc.

[16] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. (1983).Optimization by Simulated Annealing, Vol. 220, No. 4598, 1983, 671–680.

[17] Metropolis, A., Rosenbluth, W., Rosenbluth, M.N., Teller, H., Teller, E. Equation of State Calculations by Fast Computing Machines The Journal of Chemical Physics, Vol. 21, No. 6, 1953, 1087–1092.

[18] Price K.(1999). An Introduction to Differential Evolution, In: New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover,(Ed.), p. 79-108, McGraw-Hill, London, UK.

[19] Zelinka, Ivan (2004). Self Organizing Migrating Algorithm, In: New Optimization Techniques in Engineering, G. C. Onwubolu, B.V. Babu, (Ed.), Springer-Verlag.

[20] Hargis, PJ et al (1994). The Gaseous Electronics Conference Radiofrequency Reference Cell – A Defined Parallel-Plate Radiofrequency System For Experimental And Theoretical-Studies Of Plasma-Processing Discharges, Review of Scientific Instruments, 65(1) pp. 140-154, 1994

[21] Lorenz E.N. Deterministic nonperiodic flow, Journal of the Atmospheric Sciences, 20 (2), 130–141, 1963

[22] Stewart I. The Lorenz attractor exists, Nature, 406, 948–949, 2000

[23] May R. Simple mathematical model with very complicated dynamics, Nature, 261, 45-67, 1976

[24] Gilmore R., Lefranc M. The Topology of Chaos: Alice in Stretch and Squeezeland (Wiley-Interscience, New York), 2002

[25] Schuster H. G., Handbook of Chaos Control (Wiley-VCH, New York), 1999

[26] Zelinka I., Nolle L., Plasma reactor optimizing using differential evolution, in Differential Evolution: A Practical Approach to Global Optimization, Eds.: Price K.V., Lampinen J., Storn R. (Springer-Verlag, New York), 499-512, 2005

[27] Nolle L., Zelinka I., Hopgood A. A., Goodyear A., Comparison of an self organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning, Advances in Engineering Software, 36 (10), 645-653, 2005

[28] Schuster H. G., Handbook of Chaos Control (Wiley-VCH, New York), 1999

[29] Controlling Chaos and Bifurcations in Engineering Systems (CRC Press, Boca Raton), 2000
[30] Richter H., Reinschke K. J., Optimization of local control of chaos by an evolutionary algorithm, Physica D, 144, 309-334, 2000
[31] Richter H., Reinschke Kurt J., Optimization of local control of chaos by an evolutionary algorithm. Physica D 144 (2000), 309-334. 2000
[32] Richter H., An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions. In: Parallel Problem Solving from Nature-PPSN VII. (Eds.: Merelo Guerváss, J.J.; Panagiotis, A.; Beyer, H.G.; Fernández Villacanas, J.L.; Schwefel, H.P.), Lecture Notes in Computer Science, Vol. 2439, Springer-Verlag, Berlin Heidelberg New York, 2002, 308-317, 2002
[33] Zelinka I., Investigation on realtime deterministic chaos control by means of evolutionary algorithms, Proc. First IFAC Conference on Analysis and Control of Chaotic Systems, Reims, France, 211-217, 2006
[34] Nolle, L., Goodyear, A., Hopgood, A.A., Picton, P.D., Braithwaite, N.StJ.: On Step Width Adaptation in Simulated Annealing for Continuous Parameter Optimisation, Reusch, B. (Ed): Computational Intelligence - Theory and Applications, Lecture Notes in Computer Science, Vol. 2206, Springer, 2001, pp 589-598
[35] Nolle L., Zelinka I., Hopgood A. A., Goodyear A., Comparison of an self organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning, Advances in Engineering Software, 36 (10), 645-653, 2005
[36] Zelinka I., Nolle L., Plasma reactor optimizing using differential evolution, in Differential Evolution: A Practical Approach to Global Optimization, Eds.: Price K.V., Lampinen J., Storn R. (Springer-Verlag, New York), 499-512, 2006
[37] Zelinka I., Investigation on Evolutionary Deterministic Chaos Control, IFAC, Prague 2005a
[38] Zelinka Ivan, Investigation on Evolutionary Deterministic Chaos Control – Extended Study, In: 19th International Conference on Simulation and Modeling, (ECMS 2005), Riga, Latvia, in June 1-4, 2005
[39] Zelinka I., Senkerik R., Navratil E., Investigation on Evolutionary Optimization of Chaotic System, CHAOS, SOLITONS, FRACTALS, doi:10.1016/j.chaos.2007.07.045, 2007
[40] Hu G., Xie F., Xiao J., Yang J., Qu Z., Control of patterns and spatiotemporal chaos and its application, in Handbook of Chaos Control, Ed.: Schuster H. G. (Wiley VCH, New York), 2005
[41] Zelinka, I., Real-time deterministic chaos control by means of selected evolutionary algorithms Engineering Applications of Artificial Intelligence (2008), doi:10.1016/j.engappai.2008.07.008
[42] Zelinka I., Davendra D., Senkerik R., Jasek R., Oplatkova Z., Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures, Evolutionary Algorithms, Prof. Eisuke Kita (Ed.), ISBN: 978-953-307-171-8, InTech 2011
[43] Zelinka I., Davendra D., Senkerik R., Jasek R. Evolutionary Techniques And Its Possibility To Identify Catastrophic Events – An Extended Study. In Matousek Radek: Mendel 2011 -17th International Conference on SoftComputing. Brno: Brno University of Technology. 2011. ISBN 978-80-214-4302-0. p. 73-79.
[44] Alligood K, Šauer T, Yorke J, Chaos - an introduction to dynamical systems. Springer. New York