INVESTIGATION ON EVOLUTIONARY DETERMINISTIC CHAOS CONTROL

Ivan Zelinka

Institute of Process Control and Applied Informatics,
Faculty of Technology, Nam. T.G.M. 275
Tomas Bata University in Zlín
Czech Republic
Email: zelinka@ft.utb.cz

Abstract: This contribution introduces an investigation on deterministic spatiotemporal chaos control based on evolutionary algorithms use. Two evolutionary algorithms are used for chaos control here: differential evolution and self-organizing migrating algorithm. Like a model of spatiotemporal chaos so called coupled map lattices are used. Main aim of this investigation was to show that evolutionary algorithms are capable of deterministic chaos control when cost function is properly defined. Investigation consists of four different case studies with increasing calculation complexity. For both algorithms each simulation was 50 times repeated to show and check robustness of used methods. *Copyright © 2005 IFAC*

Keywords: spatiotemporal chaos, coupled map lattices, evolution, optimisation, differential evolution, SOMA.

1. INTRODUCTION

The term *deterministic chaos control* (DCC) was first coined by Ott E., Greboki C., Yorke J.A. in (Ott et all, 1990). It represents a process in which such control law is derived and used so that originally chaotic process would stabilize itself on constant level of output value or in n-periodic cycle. Since the first experiments of DCC many methods how to derive control law were developed and based on the first one (Ott et all, 1990). There are for example: pole placement (Greboki, Lai, 1999) and delay feedback (Just, Lai, 1999). Many of published methods were (originally developed for classic DCC) adapted for so called spatiotemporal chaos represented by coulpled map lattices (CML), given by (1). Models of this kind are based on set of spatiotemoporal (for 1D, Fig. 1) or spatial (for 2D, Fig. 2) cells which represents apropriate state of system elements. Typical example is CML based on so called logistic equation (Hilborn 1994, Chen 2000), which is used to simulate behaviour of system which consists of *n* mutually joined cells – logistic equations.

$$x_{n+1}(i) = (1-\varepsilon) f(x_n(i)) + \frac{\varepsilon}{2}(f(x_n(i-1)) + f(x_n(i+1))) \qquad (1)$$

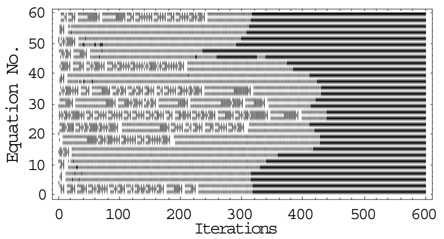

Fig. 1 1D CML with pattern T1S2

Control laws derived for CML controlling are usually based on knowledge of existing system structure (Schuster, 1999), or on use of an external observer (Chen, 2000). Main aim of this participation is to show that evolutionary algorithms (EA) are capable to control (as was also shown in (Hendrik, 2000), (Hendrik, 2002), (Hendrik, 2000a)) deterministic chaos and also CML as well as deterministic methods without internal system knowledge and operate with CML as with blackbox. Ability of EAs to succesfully work with problem kind of blackbox was many times proved, see for example realtime control of plasma reactor (Zelinka, Nolle, 2004).

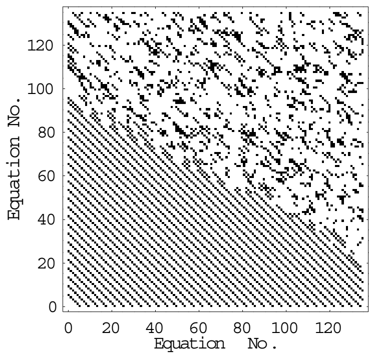

Fig. 2 2D CML with pinning imported through lattice on position (0,0). Resulting control pattern (left) is visible as well as spatiotemporal chaos (right)

2. PROBLEM DESIGN

2.1 Problem selection and case studies

The class of CML problems chosen for this comparative study was based on case studies reported in (Schuster, 1999). In general, CML control means setting of such pinnings (control CML sites) and their pining values (control values) so that system stabilizes itself on expected spatiotemporal pattern. CML as an object of study was chosen because it shows chaotic behaviour and its level of complexity can be quite rich. Investigation consists of four parts in increasing order from calculational complexity point of view and was based on paper (Hu et all, 1999). The first one is focused on pinig values estimation for a priori given pinning sites. In the second one pinning sites with a priori given pinning values were estimated by EA. The third simulation was enlargement of the previous simulation – EA was used to find minimal number of pinning sites and the fourth simulation was focused on mutual estimation of pinning sites and values, i.e. EA was searching for the minimal number of pinning sites and optimal (as much as possible) pinning values. All simulations were based on the same model and 50 times repeated for each EA with new initial conditions for each simulation. In total 400 independent simulations of spatiotemporal DCC were carried out.

2.2 The Cost Function

The fitness (cost function) has been calculated according to using the distance between desired CML state and actual CML output (2). The minimal value of this cost function, guarantee 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 (3) was used. It is synthesized from cost function (2) so that two penalty terms are added. The first one (“*p1*”) represents number of pinning sites in CML. The second one (“*p2*”) is added here to “*attract attention*” of evolutionary process on main part of cost function. If this would not be done, then mainly *p1* would be optimized and results would not be acceptable (proved by simulations). Indexes *i* and *j* are coordinates of lattice element, i.e. $CML_{i,j}$ is ith site (equation) in jth iteration. All simulations $TS_{i,j}$ was set to 0.75, i.e. CML behaviour was controlled to this state.

$$f_{cost} = \sum_{i=1}^{10} \sum_{j=80}^{100} \left| TS_{i,j} - CML_{i,j} \right|^2$$

$$\begin{aligned} &TS_{i,j} \text{ - target state of CML} \qquad (2) \\ &CML_{i,j} \text{ - actual state of controlled CML} \end{aligned}$$

$$f_{cost} = p1 + \left(p2 + \sum_{i=1}^{10} \sum_{j=80}^{100} \left| TS_{i,j} - CML_{i,j} \right| \right)^2$$

$$\begin{aligned} &TS_{i,j} \text{ - target state of CML} \\ &CML_{i,j} \text{ - actual state of controlled CML} \qquad (3) \\ &p1 \text{ - number of actually selected pinning sites} \\ &p2 \text{ - 100, heuristically set weight constant} \end{aligned}$$

2.3 Used Algorithm and Parameter Setting

For the experiments described here, stochastic optimisation algorithms, such as Differential Evolution (DE) (Price, 1999) and Self-Organizing Migrating Algorithm (SOMA) (Zelinka, 2004), had been used. Alternative algorithms, like Genetic Algorithms (GA) and Simulated Annealing (SA), are now in process, and results are hoped to be presented soon.

Differential Evolution (Price, 1999) is a population-based optimization method that works on real-number coded individuals. For each individual $\mathbf{x}_{i,G}$ in the current generation *G*, DE generates a new trial individual $\mathbf{x'}_{i,G}$ by adding the weighted difference between two randomly selected individuals $\mathbf{x}_{r1,G}$ and $\mathbf{x}_{r2,G}$ to a third randomly selected individual $\mathbf{x}_{r3,G}$. The resulting individual $\mathbf{x'}_{i,G}$ is crossed-over with the original individual $\mathbf{x}_{i,G}$. The fitness of the resulting individual, referred to as perturbated vector $\mathbf{u}_{i,G+1}$, is then compared with the fitness of $\mathbf{x}_{i,G}$. If the fitness of $\mathbf{u}_{i,G+1}$ is greater than the fitness of $\mathbf{x}_{i,G}$, $\mathbf{x}_{i,G}$ is replaced with $\mathbf{u}_{i,G+1}$, otherwise $\mathbf{x}_{i,G}$ remains in the population as $\mathbf{x}_{i,G+1}$. Deferential Evolution is robust, fast, and effective with global optimization ability. It does not require that the objective function is differentiable, and it works with noisy, epistatic and time-dependent objective functions.

SOMA is a stochastic optimization algorithm that is modelled on the social behaviour of co-operating individuals (Zelinka, 2004). It was chosen because it has been proved that the algorithm has the ability to converge towards the global optimum (Zelinka, 2004). SOMA works on a population of candidate solutions in loops called *migration loops*. The population is initialized randomly distributed over the search space at the beginning of the search. In each loop, the population is evaluated and the solution with the highest fitness becomes the leader *L*. Apart from the leader, in one migration loop, all individuals will traverse the input space in the direction of the leader. Mutation, the random perturbation of individuals, is an important operation for evolutionary strategies (ES). It ensures the diversity amongst the individuals and it also provides the means to restore lost information in a population. Mutation is different in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. This parameter has the same effect for SOMA as mutation has for GA.

The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.
The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension. An individual will travel a certain distance (called the path length) towards the leader in *n* steps of defined length. If the path length is chosen to be greater than one, then the individual will overshoot the leader. This path is perturbed randomly. For an exact description of use of the algorithms see (Price, 1999) for DE and (Zelinka, 2004) for SOMA.

The control parameter settings have been found empirically and are given in Table 1 (SOMA) and Table 2 (DE). 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 (parameter PopSize for SOMA, NP for DE). Individual length represents number of optimized parameters (number of pining sites, values, …).

Table 1: SOMA setting for case studies A, B, C & D

	A	B	C	D
PathLength	3	3	3	3
Step	3	3	3	3
PRT	0.1	0.1	0.1	0.1
PopSize	20	20	20	20
Migrations	10	10	10	10
MinDiv	0.1	0.1	0.1	0.1
Individual Length	1	10	10	20
CF Evaluations	1900	1900	1900	1900

Table 2: DE setting for case studies A, B, C & D

	A	B	C	D
NP	20	20	20	20
F	0.8	0.8	0.8	0.8
CR	0.2	0.2	0.2	0.2
Generations	100	100	100	100
Individual Length	1	10	10	20
CF Evaluations	2000	2000	2000	2000

3. EXPERIMENTAL RESULTS

Both algorithms (SOMA, DE) have been applied 50 times in order to find the optimum of all CML DCC problems. The primary aim of this comparative study is not to show which algorithm is better and worst, but to show that evolutionary DCC (EDCC) can be really used for different problems of spatiotemporal chaos control based at least on CML. Outputs of all simulations are depicted in Fig. 4-21. Fig. 4-17 show results of all 50 simulations for each case study. Fig. 18 shows a mutual comparison of algorithm performance in the point of view of the number of estimated pinning sites.

3.1 Case study A - Pinning Value Estimation.

In this case study SOMA and DE were used to estimate pining value for CML. Pining sites were a priori set according to (Hu et all, 1999). Estimated pinning value was used for all a priori defined pinning sites (each odd). Calculation was 50 times repeated and from the last population in each simulation was recorded the best, the worst and the average result (individual). All fifty triplets (best, worst, average) were used to create Fig. 4 and 5. For results verification dependance of cost value (according to (2)) on pining value was calculated and is depicted in Fig. 3. Optimal pining values are in interval 2.1 – 3.6 (cost value is 0, i.e. minimal difference between CML behaviour and desired behaviour). Based on Fig. 4 and 5 it can be stated that in all simulations were suitable pining values estimated because according to (Hu et all, 1999) suitable pining value (equal to 3) was used and here in each simulation the best values are around 2.5 and average values around 2.9.

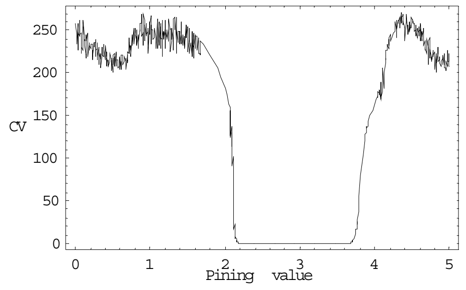

Fig. 3 Dependance of costvalue on pinning values

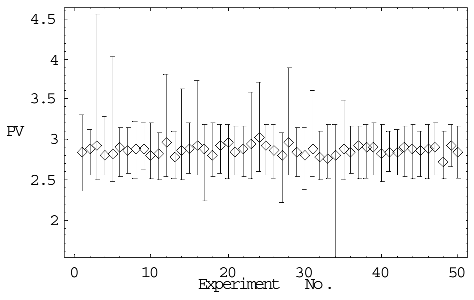

Fig. 4 Estimated pinning values by SOMA

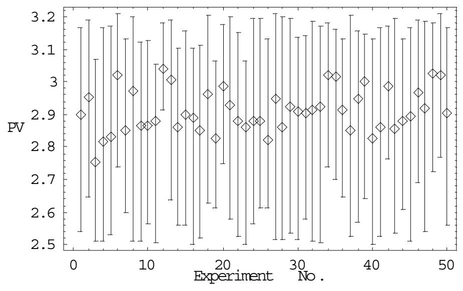

Fig. 5 Estimated pinning values by DE

3.2 Case study B - Pinning Sites Position Estimation.

Based on results from previous case study, this case study B was designed. SOMA and DE were used to estimate pining sites for CML. Pining values were a priori set equal 3 for all estimated sites according to (Hu et all, 1999). Calculation was again 50 times repeated and the best solution (pining sites) from each simulation was used to create Fig. 6 and 8. Columns represents the best solution from actual simulation and black squares in columns represent active input for pining and white squares nonused inputs, i.e. inputs without pinings.

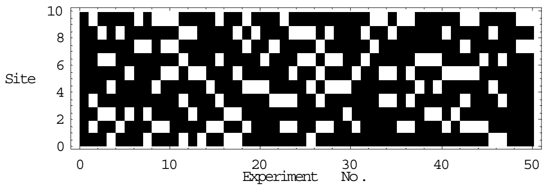

Fig. 6 Estimated pinning sites by SOMA

For better visibility of reached results also histogram (Fig. 7 and 9) was created, showing frequency of estimated pining sites. As Fig. 6 and 8 show, it can be stated that in both cases (SOMA, DE) were estimated redundant pining sites, because according to (Hu et all, 1999) certainly it is enough if pinings are at each odd (or even) site (equation, pining input). To improve this, a case study C was designed.

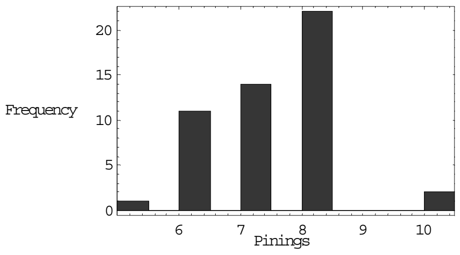

Fig. 7 Histogram of estimated pinning sites by SOMA

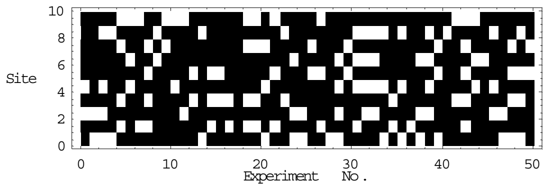

Fig. 8 Estimated pinning sites by DE

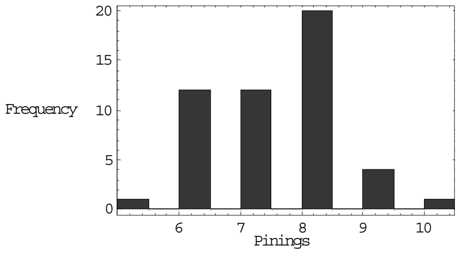

Fig. 9 Histogram of estimated pinning sites by DE

3.3 Case study C - Minimal Pinning Sites Position Estimation.

This case study was designed to improve previous results from case study B. Cost function (2) was modified to (3) as described in section 2.2. All other conditions were kept the same. Again the same figures were created (pinning sites - Fig. 10 and 12, histograms – Fig. 11 and 13). Fig. 10 and 12 show surprisingly nice structure of pinning sites. Both algorithms had found in all 50 simulations pinning sites which are on odd sites or on even sites, which is in excellent coincidence with results from (Hu et all, 1999). Because CML used here had so called cyclic boundary ((Hu et all, 1999), x(L+1) = x(1)), then it can be stated that all these solutions are equal. Based on histograms (Fig. 11 and 13) can be also made conclusion that both algorithms had almost the same performance.

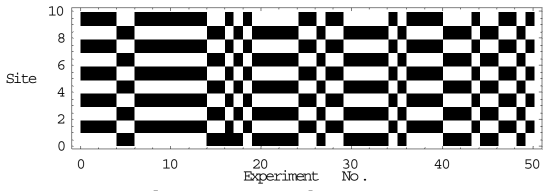

Fig. 10 Estimated pinning sites by SOMA

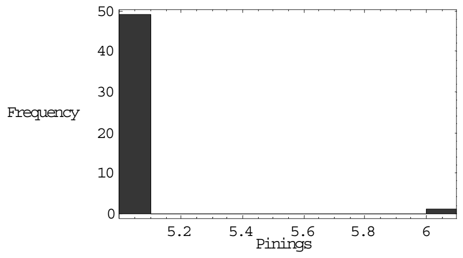

Fig. 11 Histogram of estimated pinning sites by SOMA

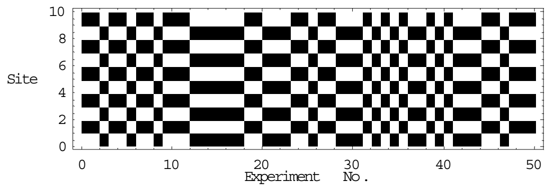

Fig. 12 Estimated pinning sites by DE

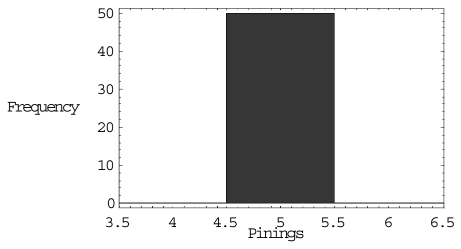

Fig. 13 Histogram of estimated pinning sites by DE

3.4 *Case study D - Minimal Pinning Values and Sites Position Estimation.*

The last case study was dedicated to estimation of minimal number of pinning sites and different pinning values. Comparing with previous case studies, for each estimated pinning site a unique pinning value was estimated here. All simulations were repeated under the same conditions as in the case study C and the same kind of figures (Fig. 14 and 16, Fig. 15 and 17) was created. In Fig. 14 and 16 are again observable pinning patterns showing that both algorithms have found the same solution more times.

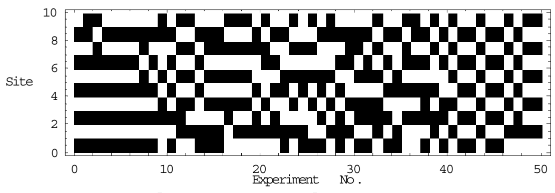

Fig. 14 Estimated pinning sites by SOMA

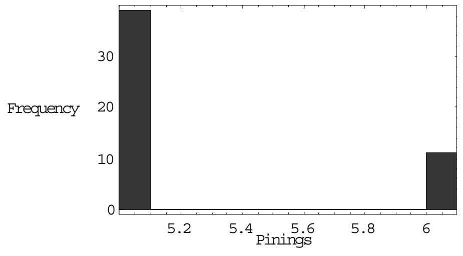

Fig. 15 Histogram of estimated pinning sites by SOMA

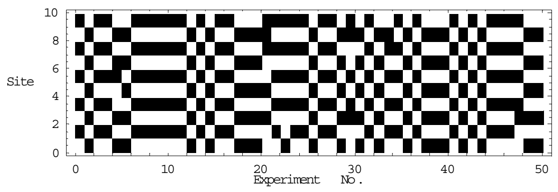

Fig. 16 Estimated pinning sites by DE

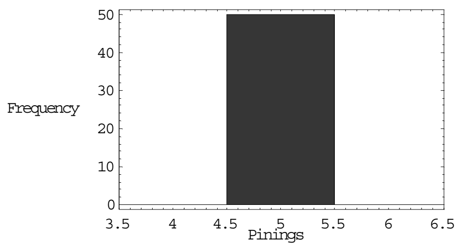

Fig. 17 Histogram of estimated pinning sites by DE

4. SELECTED MUTUAL COMPARISON

Complexity of all four case studies has increased. Based on data from all simulations two comparison can be done. The first one is from pining sites point of view. As is depicted in Fig. 18 both algorithms are comparable in performance (with small deviations). It is also visible that changes in cost functions can significantly improve estimated solutions (case C/D).

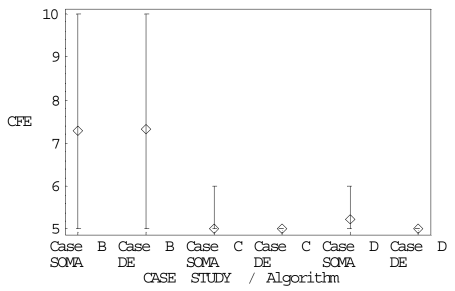

Fig. 18 Number of used pinning sites in cases B, C and D

To check that estimated pinning sites and pinning values really cause that CML will be stabilised, 400 figures were generated (4 cases × (50 + 50 simulations)) on data from all simulations. In all 400 simulations CML was stabilised on desired behaviour. For comparison with deterministic CML control (Hu et all, 1999) Fig. 19 was created and as a typical example of EDCC Fig. 20 and Fig. 21 are also depicted here.

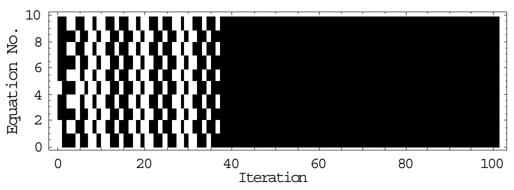

Fig. 19 Control by deterministic control law

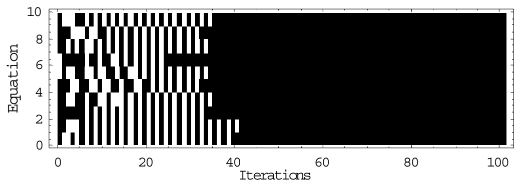

Fig. 20 Control by SOMA

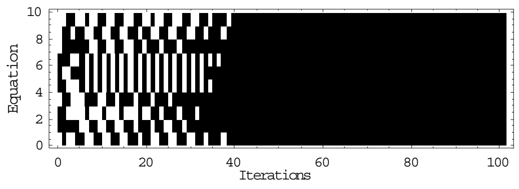

Fig. 21 Control by DE

5. CONCLUSIONS

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 was tested with 2 evolutionary algorithms – SOMA and DE) it can be stated that evolutionary deterministic chaos control is capable to solve class of CML deterministic chaos control problems. The main aim of this paper 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 50 times repeated and all 400 results (50 simulations for each algorithm and for each problem) were used to create graphs for evolutionary deterministic chaos control performance evaluation. For the comparative study two algorithms were used - DE (Price, 1999) and SOMA (Zelinka, 2004). 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. As a conclusion the following statements are presented:

1. **Reached results.** Based on results reported in Fig. 4 - 21 it can be stated that all simulations give satisfactory results and thus evolutionary deterministic chaos control is capable of solving this class of problems.
2. **Mutual comparison.** When comparing both algorithms, then it is visible that both algorithms give good results. Parameter setting for both algorithms was based on heuristically approach and thus there is a possibility that better settings can be found there.

Future research is one of the key activities in the frame of evolutionary deterministic chaos control. According to all results obtained during time it is planned that the main activities would be focused expanding of this comparative study for genetic algorithms and simulated annealing. More complicated patterns are also planned to be controlled like T1S2, etc.

ACKNOWLEDGEMENT

This work was supported by grant of the Grant Agency of the Czech Republic GACR 102/03/0070 and GACR 202/03/0789.

REFERENCES

Greboki C., Lai Y.C. Controlling Chaos, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999

Guanrong Chen, Controlling Chaos and Bifurcations in Engineering Systems, CRC Press, ISBN 0-8493-0579-9, 2000

Hendrik Richter & Kurt J. Reinschke: Optimization of local control of chaos by an evolutionary algorithm. Physica D144 (2000), 309-334.

Hendrik Richter & Kurt J. Reinschke: Optimization of local control of chaos by an evolutionary algorithm. Physica D144 (2000a), 309-334.

Hendrik Richter: An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions. In: Parallel Problem Solving from Nature-PPSN VII. (Eds.: Merelo Guervós, 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

Hilborn R.C.1994, Chaos and Nonlinear Dynamics, Oxford University Press, ISBN 0-19-508816-8, 1994

Hu G., Xie F., Xiao J., Yang J., Qu Z., Control of Patterns and Spatiotemporal Chaos and its Application, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999

Just W., Principles of Time Delayed Feedback Control, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999

Ott E., Greboki C., Yorke J.A., Controlling Chaos, Phys. Rev. Lett. 64, 1196, (1990)

Price K. 1999, An Introduction to Differential Evolution, in New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover, Eds., s. 79–108, McGraw-Hill, London, UK, 1999. ISBN 007-709506-5

Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999

Zelinka I., Nolle L. (2004), „Plasma Reactor Optimizing Using Differential Evolution", In: Price K.V., Lampinen J., Storn R., Differential Evolution : Global Optimization for Scientists and Engineers, Springer-Verlag, in print

Zelinka Ivan, 2004, SOMA – Self Organizing Migrating Algorithm",Chapter 7, 33 p. in: B.V. Babu, G. Onwubolu (eds), New Optimization Techniques in Engineering, Springer-Verlag, ISBN 3-540-20167X