

Investigation on Evolutionary Deterministic Chaos Control – Extended Study

Ivan Zelinka
Institute of Process Control and Information Technologies
Faculty of Technology
Tomas Bata University in Zlin
Mostni 5139 Zlin, Czech Republic
zelinka@ft.utb.cz

KEYWORDS

spatiotemporal chaos, coupled map lattices, evolution, optimisation, SOMA, differential evolution, genetic algorithm

ABSTRACT

This contribution presents the results of an investigation on deterministic spatiotemporal chaos control by means of evolutionary algorithms. Three evolutionary algorithms are used for chaos control: differential evolution, self-organizing migrating algorithm and genetic algorithm. Models of spatiotemporal chaos, so called coupled map lattices, are used. The main aim of this investigation was to show that evolutionary algorithms are capable of deterministic chaos control when the cost function is properly defined. The investigation consists of four different case studies with increasing calculation complexity. For each algorithm 50 simulations were carried for each problem to verify and to demonstrate the robustness of the methods used.

1 INTRODUCTION

The term *deterministic chaos control* (DCC) was first coined by Ott E., Greboki C., Yorke J.A. in (Ott E., Greboki C., Yorke J.A., 1990). It is the process of deriving and applying a control law, so that the originally chaotic process would stabilize itself either on a constant level of output values, or in an n -periodic cycle. Since the introduction of DCC, many methods for deriving control laws were developed, based on the original method (Ott E., Greboki C., Yorke J.A., 1990), for example pole placement (Greboki C., Lai Y.C. 1999) and delay feedback (Just W., 1999). Many of the published methods, which were originally developed for the classic DCC, were adapted for the so called spatiotemporal chaos, which is represented by coupled map lattices (CML), given by (1). Models of this kind are based on sets of spatiotemporal (for 1D, Figure 1) or spatial (for 2D, Figure 2) cells, which represent appropriate states of system elements. A typical example is CML based on the logistic equation, (Hilborn R.C.1994), (Guanrong Chen, 2000) which is used to simulate the behaviour of systems which consist of n mutually joined cells – logistic equations.

Control laws derived for CML controlling usually rely on existing knowledge about the system structure (Schuster H.G., 1999), or on the use of an external observer (Guanrong Chen, 2000).

The main aim of this research was to show that evolutionary algorithms (EAs) are capable of controlling CML (as it was previously shown for temporal DCC in (Hendrik Richter & Kurt J. Reinschke, 2000), (Hendrik Richter, 2002), (Hendrik Richter & Kurt J. Reinschke 2000a)) as well as deterministic methods, without the need of internal system knowledge, and that operates with CML as with a black box. The ability of EAs to successfully work with black box problems was demonstrated many times, for example for real-time control of plasma reactors (Zelinka I., Nolle L., 2005).

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

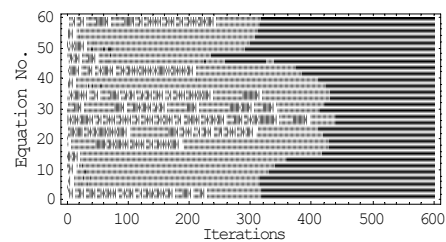


Figure 1 1D CML with pattern T1S2

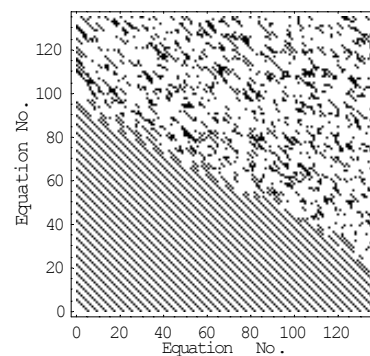


Figure 2 2D CML with pinning imported through lattice on position (0,0). The resulting control pattern (left) is visible as well as spatiotemporal chaos (right)

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 H.G., 1999). In general, CML control means setting of such pinnings (control CML sites) and their pinning values (control values) so that a system stabilizes itself

on expected spatiotemporal patterns. CML as an object of study was chosen because it shows chaotic behavior and its level of complexity can be quite rich. This research consists of four parts, presented in increasing order, from the calculational complexity point of view, and was based on the work of (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999) and (Zelinka Ivan, 2005). The first one is focused on pinning 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 an enlargement of the previous simulation – EA was used to find the 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 (i.e. as many as possible) pinning values. All simulations were based on the same CML model and were repeated 50 times for each EA with new initial conditions for each simulation. In total there were 600 independent simulations of spatiotemporal DCC where carried out.

2.2 The Cost Function

The fitness (cost function) has been calculated using the distance between the desired CML state and the actual CML output (2). The minimal value of this cost function, representing the best solution, is 0. The aim of all the simulations based on (2) 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 remaining two case studies the cost function (3) was used. It is synthesised from the cost function (2) so that two terms are added. The first one (“ $p1$ ”) represents the number of pinning sites in CML. The second one (“ $p2$ ”) is added here to “*attract attention*” of the evolutionary process on the main part of the cost function. It is in fact the number of used pinning sites, i.e. in the optimal case cost function (3) should return $p1$. If the second part of (3) would not be present, then mainly $p1$ would be optimised, so the results would not be acceptable (proved by simulations). Indexes i and j are coordinates of lattice elements, i.e. $CML_{i,j}$ is i^{th} site (equation) in j^{th} iteration. In all simulations for the target of control, $TS_{i,j}$ was set to 0.75, i.e. CML behavior was controlled to this simplest state.

$$f_{\text{cost}} = \sum_{i=1}^{10} \sum_{j=80}^{100} |TS_{i,j} - CML_{i,j}|^2 \quad (2)$$

$TS_{i,j}$ - target state of CML

$CML_{i,j}$ - actual state of controlled CML

$$f_{\text{cost}} = p1 + \left(p2 \sum_{i=1}^{10} \sum_{j=80}^{100} |TS_{i,j} - CML_{i,j}| \right)^2 \quad (3)$$

$TS_{i,j}$ - target state of CML

$CML_{i,j}$ - actual state of controlled CML

$p1$ - number of actually selected pinning sites

$p2$ - 1000, heuristically set weight constant

2.3 Optimisation Algorithm and Parameter Setting

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

Differential Evolution is a population-based optimization method that works on real-number coded individuals. For each individual $\vec{x}_{i,G}$ in the current generation G , DE generates a new trial individual $\vec{x}'_{i,G}$ by adding the weighted difference between two randomly selected individuals $\vec{x}_{r1,G}$ and $\vec{x}_{r2,G}$ to a third randomly selected individual $\vec{x}_{r3,G}$. The resulting individual $\vec{x}'_{i,G}$ is crossed-over with the original individual $\vec{x}_{i,G}$. The fitness of the resulting individual, referred to as perturbed vector $\vec{u}_{i,G+1}$, is then compared with the fitness of $\vec{x}_{i,G}$. If the fitness of $\vec{u}_{i,G+1}$ is greater than the fitness of $\vec{x}_{i,G}$, $\vec{x}_{i,G}$ is replaced with $\vec{u}_{i,G+1}$, otherwise $\vec{x}_{i,G}$ remains in the population as $\vec{x}_{i,G+1}$.

Differential 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 cooperating individuals (Zelinka Ivan, 2004). It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum (Zelinka Ivan, 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 the algorithms, see (Price K. 1999) for DE and (Zelinka Ivan, 2004) for SOMA.

The control parameter settings have been found empirically and are given in Table 1 (SOMA), Table 2 (DE) and Table 3 (GA). The main criterion for this setting was to keep the same setting of parameters as much as possible for all simulations and of course the same number of cost function evaluations as well as population size (parameter PopSize for SOMA and GA, NP for DE). Individual length represents number of optimised parameters (number of pinning sites, values...).

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

Table 3: GA setting for case studies A, B, C and D

	A	B	C	D
PopSize	20	20	20	20
Mutation	0.4	0.4	0.4	0.4
Generations	100	100	100	100
Individual Length	1	10	10	20
CF Evaluations	2000	2000	2000	2000

3 EXPERIMENTAL RESULTS

All three algorithms (SOMA, DE, GA) 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 used for different problems of spatiotemporal chaos control based at least on CML.

The outputs of all simulations are depicted in Figures 4-31. Figures 4-24 show the results of all 50 simulations for each case study. Figures 25-27 show 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, DE and GA were used to estimate the pinning value for CML. Pinning sites were a priori set according to (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999). Estimated pinning value was used for all a priori defined pinning sites (each odd). The simulations were repeated 50 times and from the last population in each simulation the best was recorded, together with the worst and the average result (individual). All fifty triplets (best, worst and average) were used to create Figures 4 - 6. For the verification of the results, the dependency of cost values (according to (2)) on pinning values was calculated and is depicted in Figure 3. Optimal pinning values are in the interval 2.1 – 3.6 (cost value is 0, i.e. minimal difference between CML behavior and desired behavior). Based on Figures 4, 5 and 6 it can be stated that in all simulations suitable pinning values were estimated because according to (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999) a suitable pinning value (equal to 2) was used and here in each simulation the best values are around 2.5 and the average values around 2.9.

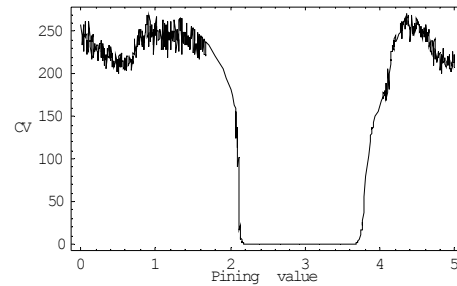


Figure 3 Dependence of costvalue on pinning values

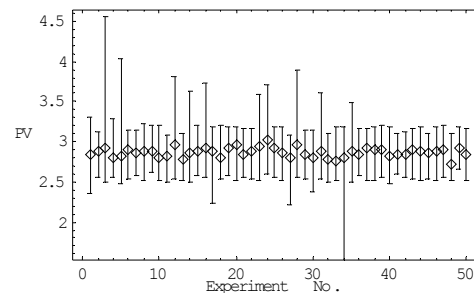


Figure 4 Estimated pinning values by SOMA

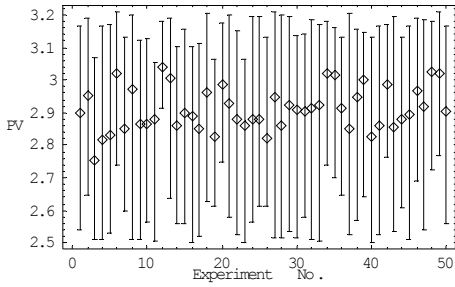


Figure 5 Estimated pinning values by DE

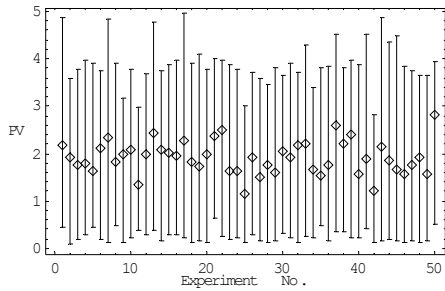


Figure 6 Estimated pinning values by GA

3.2 Case study B - Pinning Sites Position Estimation

Case study B was designed based on results from the previous case study. SOMA, DE and GA were used to estimate the pinning sites for CML. The pinning values were a priori set equal to 2 for all estimated sites according to (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999). The simulations were again repeated 50 times and the best solution (pinning sites) from each simulation was used to create Figure 7 - 12. Columns on Figures 7, 9 and 11 represent the best solution from an actual simulation and the black squares in the columns represent active input for pinning (white squares represent unused inputs, i.e. inputs without pinings).

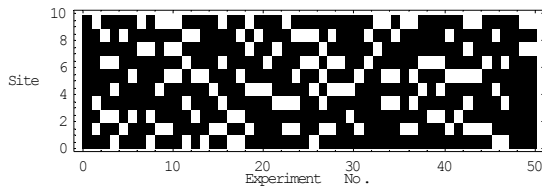


Figure 7 Estimated pinning sites by SOMA

For better visibility of the results achieved, histograms (Figure 8, 10 and 12) were also created, showing the frequency of estimated pinning sites. As the figures show, it can be stated that in two cases (SOMA, DE) there were redundant pinning sites, because according to (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999) certainly it is enough if pinings are at each odd (or even) site (equation, pinning input). GA has shown better results comparing to SOMA and DE. In order to improve SOMA and DE results, case study C was designed.

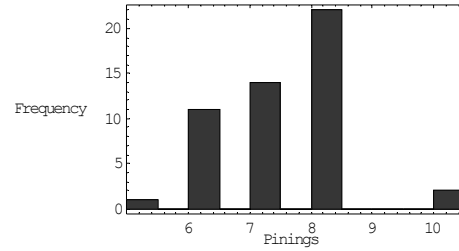


Figure 8 Histogram of estimated pinning sites by SOMA

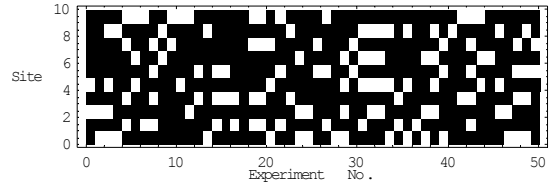


Figure 9 Estimated pinning sites by DE

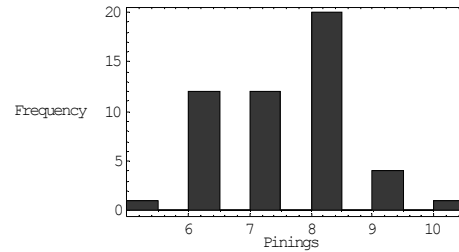


Figure 10 Histogram of estimated pinning sites by DE

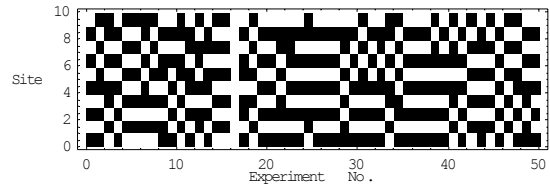


Figure 11 Estimated pinning sites by GA

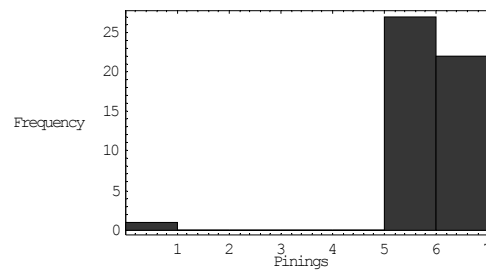


Figure 12 Histogram of estimated pinning sites by GA

3.3 Case study C - Minimal Pinning Sites Position Estimation

This case study was designed to improve the previous results from case study B. Cost function (2) was modified to (3) as described on section 2.2. All other conditions were kept the same. Again, the same types figures were created (pinning sites - Figure 13, 15 and 17, histograms - Figure 14, 16 and 18). Figures 13, 15 and 17 show surprisingly nice structures of pinning sites. SOMA and DE had found in all 50 simulations pinning

sites, which are on odd sites or on even sites, which correspond with the results from (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999). Because CML used here had so called cyclic boundary ((Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999), $x(L+1) = x(1)$), then it can be stated that all these solutions are equal. Only GA has failed in two cases (see two white columns - experiments (12,14) on Figure 17) Based on the histograms in Figures 14, 16 and 18, it can also be concluded that all algorithms demonstrated the same level of performance.

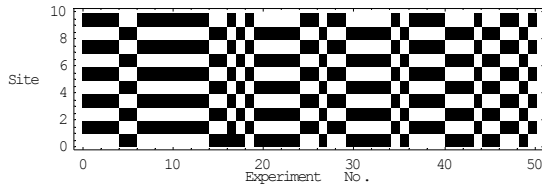


Figure 13 Estimated pinning sites by SOMA

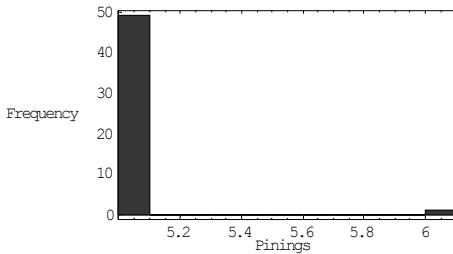


Figure 14 Histogram of estimated pinning sites by SOMA

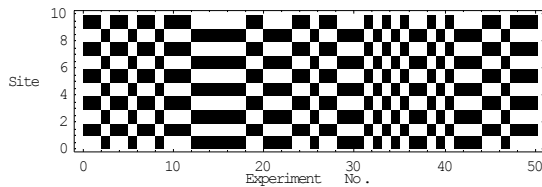


Figure 15 Estimated pinning sites by DE

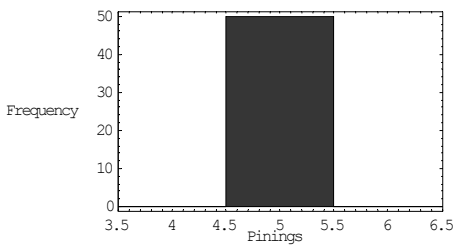


Figure 16 Histogram of estimated pinning sites by DE

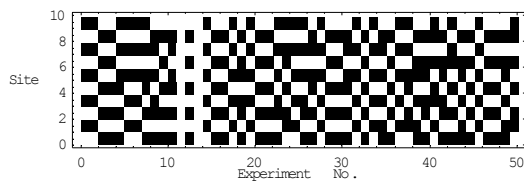


Figure 17 Estimated pinning sites by GA

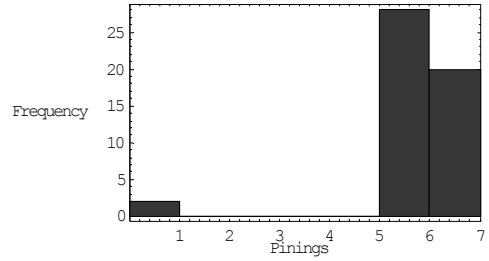


Figure 18 Histogram of estimated pinning sites by GA

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

The last case study was dedicated to the estimation of minimal numbers of pinning sites and different pinning values. In contrast to the 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 (Figure 19, 21 and 23, Figure 20, 22 and 24) was created. Figures 19, 21 show again pinning patterns demonstrating that SOMA and DE had found more times the same solution. Unexpectedly GA (Figure 23) has failed from pinning patterns point of view as well as in the four cases (experiment 5, 6, 14, 29).

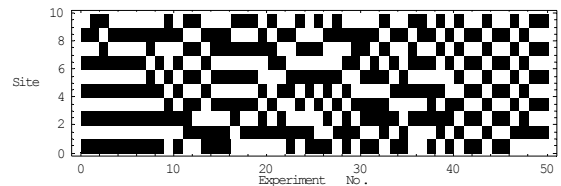


Figure 19 Estimated pinning sites by SOMA

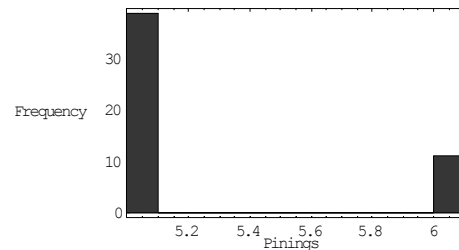


Figure 20 Histogram of estimated pinning sites by SOMA

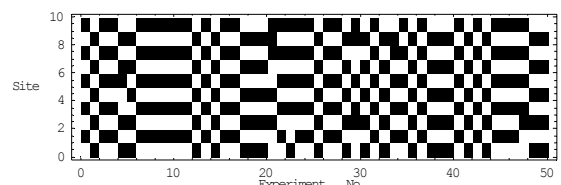


Figure 21 Estimated pinning sites by DE

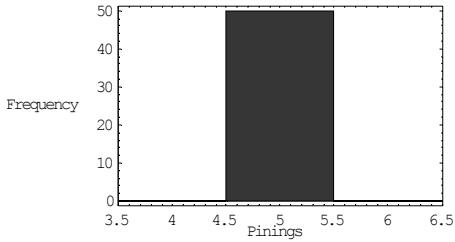


Figure 22 Histogram of estimated pinning sites by DE

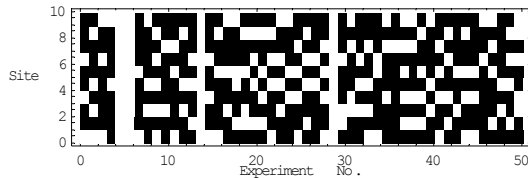


Figure 23 Estimated pinning sites by GA

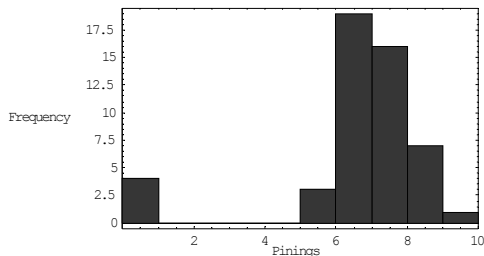


Figure 24 Histogram of estimated pinning sites by GA

4 Selected mutual comparison

The problem complexity has increased in all four case studies. Based on data from all simulations a comparison can be done from pinning sites point of view. As is depicted in Figure 25 - 27 all three algorithms are comparable in performance (with small deviations). It is also visible that changes in the cost functions can significantly improve estimated solutions (case C / D).

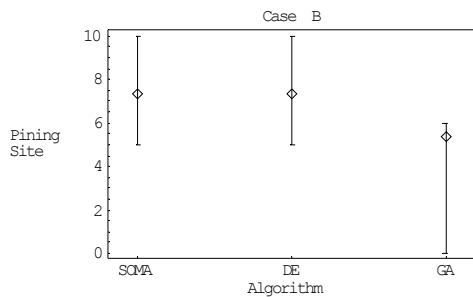


Figure 25 Mutual comparison of pinning sites - case B

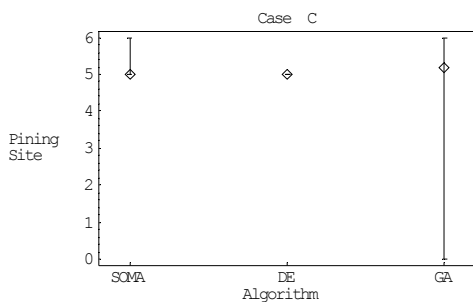


Figure 26 Mutual comparison of pinning sites - case C

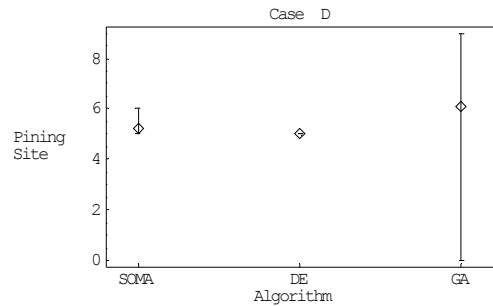


Figure 27 Mutual comparison of pinning sites - case D

In order to verify that the estimated pinning sites and pinning values really stabilize behaviour of CML, 600 figures was generated (4 cases \times (50 + 50 + 50 simulations)) on data from all simulations (including a few above mentioned wrong solutions). In all 594 = 600 - 6 simulations CML was stabilised on the desired behavior. For comparison with deterministic CML control (Hu G., Xie F., Xiao J., Yang J., Qu Z., 1999) Figure 28 was created and as a typical example of DCC and examples of ECDCC are also depicted here in Figures 29 - 31.

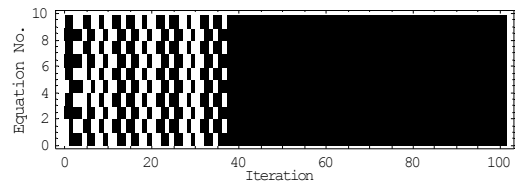


Figure 28 Control by deterministic control law

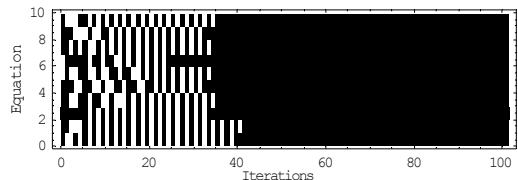


Figure 29 Control by SOMA

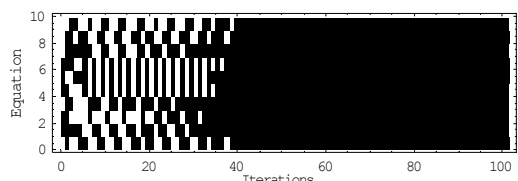


Figure 30 Control by DE

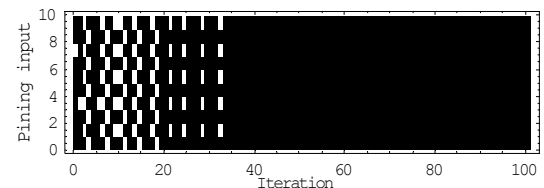


Figure 31 Control by GA

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 3 evolutionary algorithms – SOMA, DE and GA) it can be stated that evolutionary deterministic chaos control is capable to solve 1D class CML deterministic chaos control problems.

The main aim of this paper was to show how various CML control problems can be 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 600 results (50 simulations for each algorithm and for each problem) were used to create figures for performance evaluation of evolutionary deterministic chaos control.

For the comparative study three algorithms were used - DE (Price K. 1999), SOMA (Zelinka Ivan, 2004) and GA. 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 Figures 4 - 31 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. **Anomaly.** In all cases the quality of reached results has increased. Only in the case of GA an anomaly was observed, in case C and D. In case D the worst quality of pinning site “patterns” was observed comparing to case C. Also 2 simulations (C) and 4 simulations (D) have totally failed. It was probably caused by the fact that a low amount of cost function evaluations was used for EDCC and GA probably needs more Generations or / and bigger population size.
3. **Mutual comparison.** When comparing all algorithms, then it is visible that all three algorithms give good results. Parameter settings for both algorithms were based on a 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 simulated annealing and 2D CML cases. More complicated patterns are also planned to be controlled like T1S2, etc.

ACKNOWLEDGEMENT

This work was supported by grant No. MSM 7088352101 of the Ministry of Education of the Czech Republic and by grants of the Grant Agency of the Czech Republic GACR 102/03/0070.

REFERENCES

- Greboki C., Lai Y.C. 1999, Controlling Chaos, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999
- Guanrong Chen, 2000, Controlling Chaos and Bifurcations in Engineering Systems, CRC Press, ISBN 0-8493-0579-9, 2000
- Hendrik Richter & Kurt J. Reinschke, 2000: Optimization of local control of chaos by an evolutionary algorithm. *Physica D144* (2000), 309-334.
- Hendrik Richter & Kurt J. Reinschke 2000a: Optimization of local control of chaos by an evolutionary algorithm. *Physica D144* (2000), 309-334.
- Hendrik Richter, 2002: 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., 1999, 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., 1999, 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., 1990, 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., 1999, Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8, 1999
- Zelinka I., Nolle L., 2005, „Plasma Reactor Optimizing Using Differential Evolution“, In: Price K.V., Lampinen J., Storn R., *Differential Evolution* :

Global Optimization for Scientists and Engineers,
Springer-Verlag, 2005

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

Zelinka Ivan, 2005, Investigation on Evolutionary
Deterministic Chaos Control, IFAC, Prague 2005, in
print

AUTHOR BIOGRAPHIES



IVAN ZELINKA was born in Czech Republic, and went to the Technical University of Brno, where he studied technical cybernetics and obtained his degree in 1995. He obtained Ph.D. degree in technical cybernetics in 2001 at Tomas Bata University in Zlin. Now he is an associated profesor (artificial intelligence, theory of information) and head of department. His e-mail address is: zelinka@ft.utb.cz and his Web-page can be found at <http://www.ft.utb.cz/people/zelinka>.