

Tomáš Vantuch¹ · Ivan Zelinka^{2,3} · Andrew Adamatzky⁴ · Norbert Marwan⁵

Published online: 11 May 2019 © Springer Nature B.V. 2019

Abstract

Natural systems often exhibit chaotic behavior in their space-time evolution. Systems transiting between chaos and order manifest a potential to compute, as shown with cellular automata and artificial neural networks. We demonstrate that swarm optimization algorithms also exhibit transitions from chaos, analogous to a motion of gas molecules, when particles explore solution space disorderly, to order, when particles follow a leader, similar to molecules propagating along diffusion gradients in liquid solutions of reagents. We analyze these 'phase-like' transitions in swarm optimization algorithms using recurrence quantification analysis and Lempel-Ziv complexity estimation. We demonstrate that converging iterations of the optimization algorithms are statistically different from non-converging ones in a view of applied chaos, complexity and predictability estimating indicators. An identification of a key factor responsible for the intensity of their phase transition is the main contribution of this paper. We examined an optimization as a process with three variable factors—an algorithm, number generator and optimization function. More than 9000 executions of the optimization algorithms exhibit larger transition-shifting behavior while others perform rather transition-steady computing. These findings might be important for future extensions of these algorithms.

Keywords Chaos · Recurrence · Complexity · Swarm · Convergence · Phase transitions

Tomáš Vantuch tomas.vantuch@vsb.cz

> Ivan Zelinka ivan.zelinka@tdt.edu.vn; ivan.zelinka@vsb.cz

Andrew Adamatzky andrew.adamatzky@uwe.ac.uk

Norbert Marwan marwan@pik-potsdam.de

- ¹ Center ENET, Technical University of Ostrava, Ostrava, Czech Republic
- ² Modeling Evolutionary Algorithms Simulation and Artificial Intelligence, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam
- ³ Department of Computer Science, Technical University of Ostrava, Ostrava, Czech Republic
- ⁴ Unconventional Computing Lab, UWE, Bristol, UK
- ⁵ Potsdam Institute for Climate Impact Research (PIK), Member of the Leibniz Association, Potsdam, Germany

1 Introduction

Natural systems often undergo phase transition when performing a computation (as interpreted by humans), e.g., reaction-diffusion chemical systems produce a solid precipitate representing geometrical structures (Costello and Adamatzky 2017), slime mould transits from a disorderly network of 'random scouting' to prolonged filaments of protoplasmic tube connecting source of nutrients (Adamatzky 2016), 'hot ice' computer crystallizes (Adamatzky 2009). Computation at the phase transition between chaos and order was firstly studied by Crutchfield and Young (Crutchfield and Young 1988), who proposed measures of complexity characterizing the transition. The ideas were applied to cellular automata by Langton (1990): a computation at the edge of chaos occurs due to gliders. Phase transitions were also demonstrated for a genetic algorithm which falls into a chaotic regime for some initial conditions (Mitchell et al. 1993; Wright and Agapie 2001) and network traffic models (Ohira and Sawatari 1998).

Algorithmic models of evolutionary based optimization, AI and Artificial Life possess comparable features of the systems with a higher complexity they simulate (Zenil and



Gauvrit 2017; Detrain and Deneubourg 2006). We focus on the behavioral modes: the presence of random or pseudorandom cycling (analogous to gaseous phase state), ordered or stable states (analogous to solid state), or the chaotic oscillations (transitive states). Each of the modes could imply a different level of computational complexity or an algorithm performance as it was revealed on different algorithms (Boedecker et al. 2012; Bertschinger and Natschläger 2004; Kadmon and Sompolinsky 2015). By detecting such modes we can control and dynamically tune the performance of the computational systems.

A swarm-like behavior has been extensively examined in studies of Zelinka et al. (2017) where the changing dynamics of an observed algorithm was modelled by a network structure. The relevance between network features and algorithm behavior supported the control mechanism that was able to increase the algorithm performance (Tomaszek and Zelink 2016). An extensive empirical review of existing swarm-based algorithms has been brought by Schut (2010) where approaches like collective intelligence, self-organization, complex adaptive systems, multi-agent systems, swarm intelligence were empirically examined and confronted with their real models which reflected several criteria for development and verification.

Our previous study (Vantuch et al. 2018) revealed the presence of phase transitions in the computation of various swarm intelligence based algorithms. The different phases were observed on parameters estimating complexity and entropy. In the end, it was also statistically proven, that converging phases significantly differs from others in the view of mean analysis on the used parameters.

This study may serve as an advancement of our previous one. Our goal was to extend the collection of examined swarm-based algorithms, to see whether all of them performs similar transitions. On the other hand, we also extended the testing of swarm-based algorithms into other dimensions, like to rank them according to their sensitivity towards the optimization function or random number generator that drives their computation. This test is considered as necessary in order to reveal whether the phase transition occurs due to the chosen optimized function, used number generator or their appearance is clearly based on the nature of the algorithm. Having this knowledge will shape the design of the optimization algorithms towards more transitional behavior or otherwise to the stability.

2 Theoretical background

2.1 Swarm based optimization

The optimization algorithms examined in our study are representatives of bio-inspired single-objective

optimization algorithms. They iteratively maintain the population of candidates migrating through the searched space. Their current position represents the solution vector X of the optimized problem.

2.1.1 Particle Swarm Optimization

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart (1995). The main characteristic of the algorithm is the combination of the particle's aim towards the global leader and its previous best position (Kennedy and Eberhart 1995). The composition of these two stochastically altered directions modifies its current position to find a better optimum of the given function. Several reviews are available on extensions and variations of the algorithms (Banks et al. 2007; Del Valle et al. 2008).

The process of PSO starts with the initial generation of particles population P_g where g is an index of iteration. Initially, particles are distributed randomly in the searched space with a randomly adjusted vector of velocities V^g . Through the generations, all the particles in the current generation are evaluated by the given fitness function. The global leader b^g for the entire population is found by its fitness, as well as each particle keeps its personal best position p^g from his previous steps. Based on those two positions, the new velocity vector V^{g+1} for each next particles' move is derived.

$$v_i^{g+1} = wv_i^g + c_1 r_1 (b_i^g - x_i^g) + c_2 r_2 (p_i^g - x_i^g)$$
(1)

where c_1 and c_2 are the positive acceleration constants, r_1 and r_2 represent the randomly adjusted variables from the range $\langle 0, 1 \rangle$ and *w* represents the inertia weight from the range $\langle 0, 1 \rangle$.

The next generation of particles P_{PSO}^{g+1} is obtained by computing new positions X^{g+1} for each particle accordingly.

$$x_i^{g+1} = x_i^g + v_i^{g+1} \tag{2}$$

2.1.2 Self-organizing migrating algorithm

(SOMA) is a stochastic evolutionary algorithm was proposed by Zelinka (2004) and Davendra et al. (2016). Ideologically, these algorithms stand right between purely swarm optimization driven PSO and evolutionary-like DE. The entire nature of migrating individuals across the search-space is represented by steps in the defined path length and stochastic nature of a perturbation parameter that represents a specific version of the mutation. The randomness is involved through the binary vector by the adjusted extension for multi-dimensional perturbation (*PRT*) parameter [0–1] and the given formula

$$v_j^{prt} = \begin{cases} 1, & \text{if } r_j < PRT \\ 0, & \text{otherwise} \end{cases}, (j = 1, 2, \dots, d)$$
(3)

Applying V^{prt} , the path is perturbed towards a new solution using current particle and leaders position.

$$x_i^{t+1} = x_i^t + (x_L^t - x_i^t) v_i^{prt}$$
(4)

During each migration loop, each particle performs n steps according to the adjusted step size and the path length. If the path length is higher than one, the particle will travel a longer distance, that is his distance towards the leader.

2.1.3 Ant colony optimization for continuous domains

 $(ACO_{\mathbb{R}})$ (Socha and Dorigo 2008) is an extension of an algorithm inspired by ant movements firstly designed to optimize problems in a discrete domain (Dorigo and Blum 2005). This algorithm starts by initialization of the particles' positions at random places in the searched space. These positions, representing the solution candidates, are evaluated according to the optimized function and sorted by their fitness values.

$$f(X_1) \le f(X_2) \le f(X_j) \le f(X_M) \tag{5}$$

From this sorted collection, the weights *w* are calculated by the form which allows us to prefer solutions with lower fitness values. These may be in a close neighbourhood of the global optimum. Based on the position in the collection, the weights are calculated

$$w_j = \frac{1}{qM\sqrt{2\pi}} e^{-\frac{(j-1)^2}{2q^2M^2}}$$
(6)

where *q* is adjustable hyper-parameter controlling the degree on which the lower fitness values are preferred. The weights are chosen probabilistically towards the leading solution around which a new candidate solution is generated. The probability of choosing solution s_j as leading solution is given by $w_j / \sum_{a=1}^k w_a$ so that the better solutions obtain higher probability to be selected. Once a leading solution s_{lead} is chosen, the algorithm samples the neighbourhood of *i*-th real-valued component of the leading solution s_{lead}^i using a Gaussian PDF with $\mu_{lead}^i = s_{lead}^i$ and σ_{lead}^i is defined as

$$\sigma_{lead}^{i} = \xi \sum_{j=1}^{k} \frac{|s_{j}^{i} - s_{lead}^{i}|}{k - 1}$$
(7)

which stands for the average distance between the value of the *i*-th component of S_{lead} and the values of the *i*-th components of the other solutions in the archive, multiplied by a parameter *n*. The process of choosing a guiding solution and generating a candidate solution is repeated in

N times (corresponding to the number of 'ants') per iteration. Before the next iteration, the algorithm updates the solution archive keeping only the best *k* of the k + Nsolutions that are available after the solution construction process.

2.1.4 Artificial bee colony

Artificial bee colony (ABC) (Karaboga and Basturk 2007) operates with three different kinds of swarm members and with different reaction-diffusion model proposed by Tereshko (2000), Tereshko and Lee (2002), Tereshko and Loengarov (2005). The so called bees are divided into employed, onlooker bees and scout bees. The first group searches for the food around the food source, which computationally means the making use of greedy search over the available solution around the defined position.

$$v_{i,j} = x_{i,j} - \phi_{i,j}(x_{i,j} - x_{k,j})$$
(8)

where x_k is a randomly selected solution, *j* is randomly selected index within the dimension of the problem and $\phi_{i,j}$ is a random number within [-1, 1]. If the value V_i of the fitness is improved, the x_i is substituted by this found position, otherwise x_i is kept.

After all employed bees accomplish their search process, they share their positions with onlooker bees by

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{SN} \text{fit}_j} \tag{9}$$

where fit_n is the fitness value of nth solution. If the solution is not improved in a defined number of cycles, the food source is being abandoned and scout bees seek for the new source to replace using

$$x_{i,j} = lb_j - rand(0, 1) - (ub_j - lb_j)$$
 (10)

where rand(0, 1) is a generated random number from the normal distribution and *lb*, *ub* are the lower and upper boundaries of the *j*-th dimension.

2.1.5 Firefly algorithm

Firefly algorithm (FA) has been developed in 2008 by Yang and it is based on light flashing interactions of the swarm of so-called fireflies (Yang and M 2008; Yang 2010). Initially, they are distributed randomly in the searched space which is very similar compared to other swarmintelligence algorithms. The light flashing interaction represents the algorithm's novelty through the light decay caused by the increasing distance of two interacting flies, and it is defined as follows

$$\beta = \beta_0 e^{-\gamma r^2} \tag{11}$$

where β is so called the attractiveness, *r* is the distance and β_0 is the attractiveness at r = 0. The attractiveness is estimated based on a current particle's position in the searched space, so it reflects its optimization function value.

The move of the particle is than defined similarly to other optimization algorithms as

$$x_{i}^{t+1} = x_{i}^{t} + \beta_{0} e^{-\gamma r_{ij}^{2}} (x_{j}^{t} - x_{i}^{t}) + \alpha_{t} \varepsilon_{i}^{t}$$
(12)

where x_j^t represents the brightest firefly for firefly x_i^t at time t which determines its next move altered by random vector ε_i^t multiplied by randomization parameter α_t which normally decays over time as

$$\alpha_t = \alpha_0 \delta^t, \quad 0 < \delta < 1 \tag{13}$$

2.2 Number generators driving the process of optimization

All previously mentioned optimization algorithms more or less rely on a random number generator that adds some controllable amount of stochastic behavior into the process. Altering of its amount may have a critical impact on the convergence which was described and tested in available papers.

Various recent studies showed alternative options able to substitute the random number generator by other mechanisms generating numbers to drive the seek for the global optimum. In studies of Zelinka et al. (2018), the chaos number generators proved their quality in the performance increase for various solutions. These studies, therefore, underline the necessity of testing the impact of various number generators on phase transitions of the optimization algorithms.

2.3 Complexity estimation

Three indicators were selected to evaluate the current state of the system represented by swarm-based algorithm. They are the computational complexity derived by Kolmogorov complexity, predictability estimated by the Determinism and the complexity of the deterministic structure in the system represented by an Entropy. Both entropy and determinism are indicators based on recurrence quantification analysis.

2.3.1 Lempel-Ziv complexity

According to the Kolmogorov's definition of complexity, the complexity of an examined sequence X is the size of a smallest binary program that produces such sequence (Cover and Thomas 2012). Because this definition is way

too general and any direct computation is not guaranteed within the finite time (Cover and Thomas 2012), approximating techniques are often employed.

Lempel and Ziv designed a complexity (LZ complexity) estimation in a sense of Kolmogorov's definition, but limiting the estimated program only to two operations: recursive copy and paste (Lempel and Ziv 1976). The entire sequence based on an alphabet \aleph is split into a set of unique words of unequal lengths, which is called a vocabulary. The approximated binary program making use of copy and paste operations on the vocabulary can reconstruct the entire sequence. Based on the size of vocabulary (c(X)), the complexity is estimated as

$$C_{LZ}(X) = c(X)(\log_k c(X) + 1) \cdot N^{-1}$$
 (14)

where *k* means the size of the alphabet and *N* is the length of the input sequence. A natural extension for multi-dimensional LZ complexity was proposed in Zozor et al. (2005). In case of a set of *l* symbolic sequences $X^i (i = 1, ..., l)$, Lempel and Ziv's definitions remain valid if one extends the alphabet from scalar values x_k to *l*-tuples elements $(x_k^1, ..., x_k^l)$. The joined-LZC is then calculated as

$$C_{LZ}(X^1, \cdots, X^l)$$

= $c(X^1, \cdots, X^l)(\log_{k^2} c(X^1, \dots, X^l) + 1) \cdot N^{-1}.$

Conventionally LZ complexity is used to measure compressibility (Ziv and Lempel 1978; Feldman and Crutchfield 1998). Experimenting with cellular automata we found that the compressibility performs similarly well as Shannon entropy, Simpson index and morphological diversity in detecting phase transitions (Redeker et al. 2013; Adamatzky 2012). For example, in cellular automata we can detect formation of travelling localisations, propagating patterns, stable states and cycles (Adamatzky and Chua 2012; Ninagawa and Adamatzky 2014). The compressibility was also well used for the analysis of living systems, e.g., EEG signals (Bhattacharya 2000; Aboy et al. 2006) and DNA sequences (Orlov and Potapov 2004), and classification of spike trains (Amigó et al. 2004).

2.3.2 Recurrence quantification analysis

The recurrence plot (RP) is the visualization of the recurrences of *m*-dimensional system states $\mathbf{x} \in \mathbb{R}^m$ in a phase space (Marwan et al. 2007b). Recurrence is defined as closeness of these states \mathbf{x}_i (i = 1, 2, ..., N where *N* is the trajectory length), measured by thresholded pairwise distances. Formally, the RP can be expressed by $R_{i,j}(\varepsilon) = \Theta(\varepsilon - ||\mathbf{x}_i - \mathbf{x}_j||)$ with $\Theta(\cdot)$ the Heaviside step function. The Euclidean norm is the most frequently applied distance metric $|| \cdot ||$ and the threshold value ε can be chosen according to several techniques (Koebbe and Mayer-Kress

Fig. 1 Recurrence plots of the PSO (**a**–**c**), DE (**d**–**f**), and SOMA (ghi) behavior calculated as similarities among the particles' positions X_t grouped into the windows of populations P_{w_i} during their (**a**, **d**, **g**) "post-initial" (10th migration), (**b**, **e**, **h**) "top-converging" (60th migration) and (**c**, **f**, **i**) "post-converging" (400th migration) phase



1994; Zbilut et al. 2002; Zbilut and Webber 1992; Marwan et al. 2007b; Schinkel et al. 2008; Kraemer et al. 2018).

If only a one-dimensional measurement u_i of the system's dynamics is given, the phase space trajectory has to be reconstructed from the time series $\{u_i\}_{i=1}^N$, e.g., by using the time-delay embedding $\mathbf{x}_i = (u_i, u_{i+\tau}, \dots, u_{i+(m-1)\tau})$, where *m* is the embedding dimension and τ is the embedding delay (Packard et al. 1980). The parameters *m* and τ may be found using methods based on false nearest neighbors and auto-correlation (Kantz and Schreiber 1997).

The recurrence quantification (RQA) measures applied in this experiment describe the predictability and level of chaos in the observed system. Determinism is defined as the percentage of points that form diagonal lines

$$DET = \frac{\sum_{l=2}^{N} lP(l)}{\sum_{l=1}^{N} lP(l)}$$
(15)

where P(l) is the histogram of the lengths l of the diagonal lines (Marwan et al. 2007b). Its values, ranging between zero and one, estimate the predictability of the system.

The measure divergence is related to the sum of the positive Lyapunov exponents, naturally computing the amount of chaos in the system, and is defined as

$$DIV = L_{\max}^{-1}, \quad L_{\max} = \max(\{l_i; i = 1, ..., N_l\})$$
 (16)

where L_{max} is the longest diagonal line in the RP (excluding the main diagonal line) (Marwan et al. 2007b).

3 Experiment design

The motivation is to identify the key factor for the phase transitions in swarm optimization algorithms. Based on our previous study and as it was mentioned previously, we used three metrics M in order to evaluate the phase transitions,



Fig. 2 Comparative visualization of mean distributions of standard deviations of determinisms of algorithm progresses. Sub-figures show the values separated on different criteria comparing their impact: **a** applied filtering based on number generator while subfilters were

optimization algorithm and optimized function, \mathbf{b} applied filtering based on optimized function while subfilters were optimization algorithm and number generator

they are the Kolmogorov complexity $(m_1 = Kc)$, determinism $(m_2 = DET)$ and divergence $(m_3 = DIV)$. The progress of swarm optimization execution is captured as a tensor *T* which is part by part $(t_1 \dots t_N)$ examined by metrics *M*. Their changes within one optimization reflected its transitions τ . The intensity of the transition is simply evaluated as the standard deviation of the metric value $\tau_i = \text{std}(m_i(T))$.

The examined factors that may alter the significance of τ were represented by the kind of the algorithm (*A*), the number generator ($G \in \{\text{rand}, \text{chaos}, \text{order}\}$) and optimized function (*F*).

The algorithms were mentioned previously, therefore, $A \in \{\text{SOMA}, \text{PSO}, \text{FA}, \text{ABC}, \text{ACO}_{\mathbb{R}}\}$. The number of generators were considered as an important source of chaos-order transitions, so three of them were examined $(G \in \{\text{rand}, \text{chaos}, \text{order}\})$. In the first case, the standard random number generator (Mersenne Twister) (Matsumoto and Nishimura 1998) was kept to drive the optimization process, while in the *chaos* and *order* mechanisms, the numbers were loaded from time series generated by chaotic system—Lorenz attractor (Stewart 2000) (all three coordinates x, y, z were used as the source of *randomness*) and deterministic processes—the sin(x) equidistantly sampled, similarly as in Zelinka et al. (2018).

Our aim was to test the algorithms on dimensionally scalable fitness functions F having at least one global optimum surrounded by multiple local optimums. These conditions were met making use of the Rastrigin function (Eq. 17), the Rosenbrock function (Eq. 18) and Ackley's function (Eq. 19) (Abiyev and Tunay 2015),

 $F \in \{ackley, rosenbrock, rastrigin\}.$

$$f(x) = A \cdot n + \sum_{i=1}^{n} (x_i^2 - A \cdot \cos(2\pi x_i))$$
(17)

$$f(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)]$$
(18)

$$f(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_{i}^{2}}\right) - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_{i})\right) + 20 + e$$
(19)

Table 1 Determinism parameters: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Table 2 Entropy parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Ras.

Ros.

Ack.

Driver

(a) Artificial Bee colony

Driver	Ack.	Ras.	Ros.	All
(a) Artificial Bee c	olony			
Chaos	0.03	0.03	0.05	0.03
Order	0.08	0.03	0.05	0.05
Rand	0.08	0.03	0.05	0.05
All	0.07	0.03	0.05	
(b) Ant colony				
Chaos	0.14	0.12	0.1	0.12
Order	0.21	0.17	0.1	0.16
Rand	0.19	0.12	0.1	0.14
All	0.18	0.14	0.1	
(c) Particle swarm				
Chaos	0.19	0.16	0.19	0.18
Order	0.08	0.07	0.05	0.07
Rand	0.18	0.13	0.16	0.16
All	0.15	0.12	0.13	
(d) Self organizing	migrating			
Chaos	0.01	0.13	0.14	0.09
Order	0.13	0.0	0.01	0.05
Rand	0.02	0.0	0.02	0.01
All	0.05	0.05	0.05	
(e) Firefly algorithm	n			
Chaos	0.08	0.08	0.14	0.1
Order	0.03	0.08	0.09	0.08
Rand	0.03	0.09	0.11	0.07
All	0.05	0.08	0.11	

()	2			
Chaos	0.14	0.16	0.12	0.14
Order	0.21	0.16	0.12	0.16
Rand	0.21	0.15	0.13	0.16
All	0.19	0.16	0.12	
(b) Ant colony				
Driver	ack.	ras.	ros.	all
Chaos	0.55	0.56	0.34	0.49
Order	0.86	0.71	0.33	0.64
Rand	0.81	0.57	0.34	0.57
All	0.74	0.61	0.34	
(c) Particle swa	arm			
Chaos	0.48	0.44	0.48	0.47
Order	0.4	0.18	0.16	0.26
Rand	0.46	0.37	0.44	0.42
All	0.44	0.33	0.35	
(d) Self organiz	zing migrating			
Chaos	0.05	0.56	0.57	0.38
Order	0.51	0.03	0.02	0.19
Rand	0.1	0.04	0.08	0.07
All	0.21	0.2	0.22	
(e) Firefly algo	rithm			
Chaos	0.5	0.53	0.78	0.58
Order	0.1	0.52	0.2	0.31
Rand	0.08	0.51	0.35	0.31
All	0.25	0.52	0.37	

The entire experiment is, therefore, the set of several executions of the optimization process O which is always defined by those three factors (O(A, G, F)). The pseudocode of the experiment may be seen below.

```
Data: A, G, F, N = 200

Result:

for each a in A do

for each g in G do

for each f in F do

for in 1 .. N do

T<sub>i</sub> = O(a, g, f);

for each m in M do

| \tau_{i,m} = \operatorname{std}(m(T_i));

end

end

end

end
```

Algorithm 1: Iterative execution of all adjustments of the experiment

The transitions τ will differ from each other based on the adjusted factors of optimization. To identify the key factor

responsible for the increasing amplitude of τ we need to compare the means for each factor. An additional outcome will be the reveal of the conditional means for each algorithm while one of its factors will be fixed.

3.1 Tensor data obtained from the optimization

The optimization step of the optimization algorithms is represented by the positions $(X_{t_1} = \{x_{t_1,1}, x_{t_1,2}, ..., x_{t_1,D}\})$ taken by its population members $(P = \{p_1, p_2, ..., p_N\})$ during their migrations/iterations $(p_1 = X_{t_1,1}, X_{t_2,1}, ..., X_{t_m,1})$. All of them are stored for the further examination. The time windows *w* of iterations are taken and transferred into matrices of particles positions where columns are particle's coordinates and rows are ordered particles by their population number and time $(P_{w_i} = \{x_{t_i,1}, x_{t_i,2}, ..., x_{t_i,N}, x_{t_{i+1},1}, x_{t_{i+1},2}, ..., x_{t_{i+1},N}, ..., x_{t_{i+w},N}\})$.

The changes and interactions inside of their migrating populations are not usually visible in convergence plots;

All

~11

Dog

Table 3 Kolmogorov complexity parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Table 4 Determinism parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Dec

Aal

Driver	Ack.	Ras.	Ros.	All
(a) Artificial B	ee Colony			
Chaos	0.14	0.17	0.23	0.18
Order	0.24	0.16	0.23	0.21
Rand	0.25	0.16	0.23	0.21
All	0.21	0.16	0.23	
(b) Ant Colony	7			
Chaos	0.8	0.59	0.75	0.72
Order	1.36	0.98	0.71	1.01
Rand	1.3	0.71	0.61	0.87
All	1.15	0.76	0.69	
(c) Particle Sw	arm			
Chaos	1.03	1.1	1.0	1.04
Order	0.55	0.56	0.57	0.56
Rand	1.13	1.11	0.93	1.06
All	0.91	0.92	0.83	
(d) Self organiz	zing migrating			
Chaos	0.1	0.88	0.88	0.62
Order	0.89	0.02	0.03	0.31
Rand	0.16	0.1	0.13	0.13
All	0.38	0.33	0.35	
(e) Firefly algo	rithm			
Chaos	0.9	0.9	0.96	0.92
Order	0.28	0.86	0.39	0.56
Rand	0.27	0.9	0.95	0.71
All	0.52	0.88	0.73	

however, changes during the convergence may be estimated using recurrence plots. For this purpose, three selected windows of algorithms' iterations were visualized to spot the differences among them. Figure 1 illustrates how phases of the algorithm convergences are reflected in RPs.

Complexity estimation. The obtained matrix P_{w_i} served as input for a joint Lempel-Ziv complexity (LZC) estimation and RQA.

For the purpose of joint LZC estimation, the input matrix was discretized into adjustable number of letters n_l of an alphabet by the given formula. Let $p_{\min} = \min\{p_j | 1 \le j \le w\}$, $p_{\max} = \max\{p_j | 1 \le j \le w\}$ and $p_d = p_{\max} - p_{\min}$ then each element p_j is assigned value $p_j \leftarrow \lfloor n_l \frac{p_j - p_{\min}}{p_d} \rfloor$. The joint-LZC therefore stands, in our case, for the complexity of time ordered *n* dimensional tuples (populations).

In case of RQA, there is a possibility to directly use the spatial data representation (Marwan et al. 2007a), therefore

Diivei	ACK.	ixas.	103.	an
(a) Random nur	nber generator			
ABC	0.08	0.03	0.05	0.05
$ACO_{\mathbb{R}}$	0.19	0.12	0.1	0.14
FA	0.03	0.09	0.11	0.07
PSO	0.18	0.13	0.16	0.16
SOMA	0.02	0.01	0.02	0.01
All	0.1	0.07	0.08	
(b) Order numb	er generator			
ABC	0.08	0.03	0.05	0.05
$ACO_{\mathbb{R}}$	0.21	0.17	0.1	0.16
FA	0.03	0.08	0.09	0.08
PSO	0.08	0.07	0.05	0.07
SOMA	0.13	0.0	0.01	0.05
All	0.11	0.07	0.06	
(c) Chaos numb	er generator			
ABC	0.03	0.03	0.05	0.03
$ACO_{\mathbb{R}}$	0.14	0.12	0.1	0.12
FA	0.08	0.08	0.14	0.1
PSO	0.19	0.16	0.19	0.18
SOMA	0.01	0.13	0.14	0.09
All	0.09	0.11	0.12	

we did not apply the Takens' embedding theorem (Takens 1980; Marwan et al. 2015) and we directly calculated the RP from our source data. The RQA features like determinism and divergence were calculated.

4 Results

Deivor

Each of the algorithms were executed 200 times. During each execution the hyper-parameters were adjusted randomly in order to fairly examine the presence of phase transitions regardless of the optimization performance. The significance of phase transition τ was estimated by the standard deviation of the estimated complexity parameter (Determinism, Entropy and Kolmogorov complexity). The higher level of τ implies a higher level of fluctuating behavior of the optimization while minimal τ stood for a transition-less optimization.

In Fig. 2 various levels of phase transitions have been observed across all the available setups. From the given charts, there may be spotted those three defined factors affecting the significance of the phase transitions (A, G, F). Their impact is visually different, therefore our further

Table 5 Entropy parameter: calculated averages of standard devia-
tions of τ for the given setups to picture the amount of phase transition
observed during optimization

Table 6 Kolmogorov complexity parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Driver	Ack.	Ras.	Ros.	all
(a) Random nui	mber generator			
ABC	0.21	0.15	0.13	0.16
$ACO_{\mathbb{R}}$	0.81	0.57	0.34	0.57
FA	0.08	0.51	0.35	0.31
PSO	0.46	0.37	0.44	0.42
SOMA	0.1	0.04	0.08	0.07
All	0.33	0.33	0.26	
(b) Order numb	er generator			
ABC	0.21	0.16	0.12	0.16
$ACO_{\mathbb{R}}$	0.86	0.71	0.33	0.64
FA	0.1	0.52	0.2	0.31
PSO	0.4	0.18	0.16	0.26
SOMA	0.51	0.03	0.02	0.19
All	0.45	0.32	0.17	
(c) Chaos numb	per generator			
ABC	0.14	0.16	0.12	0.14
$ACO_{\mathbb{R}}$	0.55	0.56	0.34	0.49
FA	0.5	0.53	0.78	0.58
PSO	0.48	0.44	0.48	0.47
SOMA	0.05	0.56	0.57	0.38
All	0.34	0.44	0.42	

Driver	Ack.	Ras.	Ros.	All
(a) Random nur	nber generator			
ABC	0.25	0.16	0.23	0.21
$ACO_{\mathbb{R}}$	1.3	0.71	0.61	0.87
FA	0.27	0.9	0.95	0.71
PSO	1.13	1.11	0.93	1.06
SOMA	0.16	0.1	0.13	0.13
All	0.62	0.59	0.57	
(b) Order numb	er generator			
ABC	0.24	0.16	0.23	0.21
$ACO_{\mathbb{R}}$	1.36	0.98	0.71	1.01
FA	0.28	0.86	0.39	0.56
PSO	0.55	0.56	0.57	0.56
SOMA	0.89	0.02	0.03	0.31
All	0.71	0.52	0.39	
(c) Chaos numb	er generator			
ABC	0.14	0.17	0.23	0.18
$ACO_{\mathbb{R}}$	0.8	0.59	0.75	0.72
FA	0.9	0.9	0.96	0.92
PSO	1.03	1.1	1.0	1.04
SOMA	0.1	0.88	0.88	0.62
All	0.59	0.71	0.74	

examination is to reveal which of those affect the level of τ the most significantly.

Simply by filtering of one examined factor from the entire database and averaging its τ values on the given subset, we may estimate whether the optimization process is rather stable or not while applying this examined factor. To estimate how much one factor is affecting another, we need to perform another filtering on the given subset. This secondary filtering will reveal the conditional phase transition significance based on the given subfactor and it will show how this subfactor behaves on the given conditions. We started by filtering based on optimization algorithm *A* and remaining factors for subselections were *G* and *F*, therefore we were able to estimate how modification of *G* and *F* alters the behavior of *A*. Later we switched and as a main factor we selected *G* and *F* accordingly.

4.1 Selection by type of algorithm

Separation of the data set by algorithm factor A results into five different subsets where we are able to examine the influence of number generator G and optimization problem F (see Figs. 1, 2, 3). Comparing the given setups, we may observe some differences among phase transition levels. It is difficult to estimate from these charts, whether the number generator is affecting the phase transitions more than the optimized function or vice versa. What is clearly visible, and will be estimated further as well, is a significant difference among algorithms which is caused by the way how they are designed to process the computation. Among algorithms with the rather higher mean of τ , we may account PSO and ACO_R especially due to values examined by Determinism and Kolmogorov complexity estimation. On the other hand, very stable behavior may be seen in cases of SOMA and ABC examinations.

From the Tables 1, 2, and 3 it is not clear which of the secondary factors has higher influence. In cases of ABC, $ACO_{\mathbb{R}}$ and FA, the optimized function affects τ more significantly than the number generator. Differences of τ means are much higher based on *G* compare to *F*. In other cases (PSO and SOMA) the much higher influence is obtained altering the number generator, while optimization seems not to be so sensitive on changing the optimization function. These findings were spotted in all three examined metrics *M*. The difference between those subfactors is rather small, but still, we may observe that changing the

Table 7 Determinism parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Table 8 Enropy parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Driver	Chaos	Order	Rand	All
(a) Ackley fund	ction			
ABC	0.03	0.08	0.08	0.07
$\operatorname{ACO}_{\mathbb{R}}$	0.14	0.21	0.19	0.18
FA	0.08	0.03	0.03	0.05
PSO	0.19	0.08	0.18	0.15
SOMA	0.01	0.13	0.02	0.05
All	0.09	0.11	0.1	
(b) Rastrigin fu	nction			
ABC	0.03	0.03	0.03	0.03
$\operatorname{ACO}_{\mathbb{R}}$	0.12	0.17	0.12	0.14
FA	0.08	0.08	0.09	0.08
PSO	0.16	0.07	0.13	0.12
SOMA	0.13	0.01	0.01	0.05
All	0.11	0.07	0.07	
(c) Rosenbrock	function			
ABC	0.05	0.05	0.05	0.05
$\operatorname{ACO}_{\mathbb{R}}$	0.1	0.1	0.1	0.1
FA	0.14	0.09	0.11	0.11
PSO	0.19	0.05	0.16	0.13
SOMA	0.14	0.01	0.02	0.05
All	0.12	0.06	0.08	

optimization function may change the phase transition significance more likely than changing the number generator.

4.2 Selection by type of number generator

In this second view on the result data, we will filter based on the number generator G at first and than as a subfactors to compare, we will use the type of algorithm A and optimized function F. Results are depicted in Tables 4, 5 and 6. The differences on τ mean are much higher on algorithm based filtering compare to the optimized function based filtering. This simply implies that optimization procedure alters its phase transitions significance based on the kind of applied algorithm rather than optimized function. This observation was spotted in all kinds of examined metrics.

SOMA with ABC appeared as the most stable having the lowest values of average differences of complexity parameters, while PSO and $ACO_{\mathbb{R}}$ performed the exact opposite indicating the much higher presence of phase transitions in this algorithm. FA was performing rather

Driver	Chaos	Order	Rand	All
(a) Ackley funct	tion			
ABC	0.14	0.21	0.21	0.19
$ACO_{\mathbb{R}}$	0.55	0.86	0.81	0.74
FA	0.5	0.1	0.08	0.25
PSO	0.48	0.4	0.46	0.44
SOMA	0.05	0.51	0.1	0.21
All	0.34	0.45	0.33	
(b) Rastrigin fur	nction			
ABC	0.16	0.16	0.15	0.16
$ACO_{\mathbb{R}}$	0.56	0.71	0.57	0.61
FA	0.53	0.52	0.51	0.52
PSO	0.44	0.18	0.37	0.33
SOMA	0.56	0.03	0.04	0.2
All	0.44	0.32	0.33	
(c) Rosenbrock	function			
ABC	0.12	0.12	0.13	0.12
$ACO_{\mathbb{R}}$	0.34	0.33	0.34	0.34
FA	0.78	0.2	0.35	0.37
PSO	0.48	0.16	0.44	0.35
SOMA	0.57	0.02	0.08	0.22
All	0.42	0.17	0.26	

Table 9 Kolmogorov complexity parameter: calculated averages of standard deviations of τ for the given setups to picture the amount of phase transition observed during optimization

Driver	Chaos	Order	Rand	All
(a) Ackley fund	ction			
ABC	0.14	0.24	0.25	0.21
$ACO_{\mathbb{R}}$	0.8	1.36	1.3	1.15
FA	0.9	0.28	0.27	0.52
PSO	1.03	0.55	1.13	0.91
SOMA	0.1	0.89	0.16	0.38
All	0.59	0.71	0.62	
(b) Rastrigin fu	inction			
ABC	0.17	0.16	0.16	0.16
$ACO_{\mathbb{R}}$	0.59	0.98	0.71	0.76
FA	0.9	0.86	0.9	0.88
PSO	1.1	0.56	1.11	0.92
SOMA	0.88	0.02	0.1	0.33
All	0.71	0.52	0.59	
(c) Rosenbrock	function			
ABC	0.23	0.23	0.23	0.23
$ACO_{\mathbb{R}}$	0.75	0.71	0.61	0.69
FA	0.96	0.39	0.95	0.73
PSO	1.0	0.57	0.93	0.83
SOMA	0.88	0.03	0.13	0.35
All	0.74	0.39	0.57	



Fig. 3 Estimated sensitivity of the optimization algorithms' phase transitions towards used kind of fitness function and the number generator. a Standard deviation values of Determinism, b Standard deviation values of entropy and c Standard deviation values of Kolmogorov complexity

transitions occurring computations mostly on average of the observed algorithms.

4.3 Selection by type of optimized function

The filter based on the optimized function F only confirms the previously observed findings, but this time the compared subfactors are the type of algorithm A and number generator G. Results are depicted in Tables 7, 8, and 9. The differences on τ mean are significantly higher on algorithm based filtering compare to the number generator based filtering. This again implies, that optimization procedure alters its phase transitions significance due to the kind of applied algorithm more likely than optimized function. This observation was spotted in all kinds of examined metrics.

Due to our finding, the third examination of phase transitions was performed on the results filtered by algorithms' kind. We averaged the impact of the other factors for each algorithm to measure how they change alters the algorithms' behavior. The results are depicted in Fig. 3 where we can clearly observe that PSO, $ACO_{\mathbb{R}}$ and FA are the groups of algorithms with the higher level of phase transitions while SOMA and ABC are representatives of rather phase-stable optimization approaches. These results were obtained similarly on all examined complexity measures with visible correlation among them.

5 Conclusions

The varying instability of swarm optimization behavior was examined in these experiments in a slightly larger scale comparing to our first initial study (Vantuch et al. 2018).

Five swarm-intelligence based optimization algorithms were examined in nine different setups based on three different number generators and three different optimized functions. The main motivation was to compare which factor most likely affects the amount of phase transitions. From our simulations, it is clearly visible that the type of optimization algorithm is the key factor affecting the significance of phase transitions. The remaining factors were also altering this phenomenon significantly but in a much smaller scale.

The last comparison only underlines our conclusions. Algorithms were depicted in Fig. 3 where the sensitivity on the number generator (the average standard deviation on all number generators) was in all cases very close to the sensitivity on the fitness function (the average standard deviation on all fitness functions), while differences among the algorithms were very significant. All three complexity measures confirmed this observation with a slight visible correlation.

Our future work has to examine whether the phase transitions are beneficial for the convergence and which algorithm is using them this way, because otherwise, they may perform only disruptive element which is necessary to minimize. On the other hand, our results sometimes returned an outlier observations (behavior of some algorithm changed too much or not at all) which may be caused by another, not considered factor. Our future study will consider the examination of initial population distribution on the phase transition significance as well.

Acknowledgements This paper was supported by the following Projects: LO1404: Sustainable development of ENET Centre; SP2019/28 and SGS 2019/137 Students Grant Competition and the Project LTI17023 "Energy Research and Development Information Centre of the Czech Republic" funded by Ministry of Education, Youth and Sports of the Czech Republic, program INTER-EXCELLENCE, subprogram INTER-INFORM, and DFG Projects MA4759/8 and MA4759/9.

References

- Abiyev RH, Tunay M (2015) Optimization of high-dimensional functions through hypercube evaluation. Comput Intell Neurosci 2015:17
- Aboy M, Hornero R, Abásolo D, Álvarez D (2006) Interpretation of the Lempel-Ziv complexity measure in the context of biomedical signal analysis. IEEE Trans Biomed Eng 53(11):2282–2288
- Adamatzky A (2009) Hot ice computer. Phys Lett A 374(2):264-271

- Adamatzky A (2012) On diversity of configurations generated by excitable cellular automata with dynamical excitation intervals. Int J Mod Phys C 23(12):1250085
- Adamatzky A (2016) Advances in Physarum machines: sensing and computing with slime mould, vol 21. Springer, Cham
- Adamatzky A, Chua LO (2012) Phenomenology of retained refractoriness: on semi-memristive discrete media. Int J Bifurcat Chaos 22(11):1230036
- Amigó JM, Szczepański J, Wajnryb E, Sanchez-Vives MV (2004) Estimating the entropy rate of spike trains via Lempel-Ziv complexity. Neural Comput 16(4):717–736
- Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part I: background and development. Nat Comput 6(4):467–484
- Bertschinger N, Natschläger T (2004) Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput 16(7):1413–1436
- Bhattacharya J et al (2000) Complexity analysis of spontaneous EEG. Acta Neurobiol Exp 60(4):495–502
- Boedecker J, Obst O, Lizier JT, Mayer NM, Asada M (2012) Information processing in echo state networks at the edge of chaos. Theory Biosci 131(3):205–213
- Costello BDL, Adamatzky A (2017) Calculating Voronoi diagrams using chemical reactions. In: Adamatzky A (ed) Advances in unconventional computing. Springer, Cham, pp 167–198
- Cover TM, Thomas JA (2012) Elements of information theory. Wiley, New York
- Crutchfield JP, Young K (1988) "Computation at the onset of chaos," in The Santa Fe Institute. Citeseer, Westview
- Davendra D, Zelinka I et al (2016) Self-organizing migrating algorithm. In: New optimization techniques in engineering
- Del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez J-C, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12(2):171–195
- Detrain C, Deneubourg J-L (2006) Self-organized structures in a superorganism: do ants "behave" like molecules? Phys Life Rev 3(3):162–187
- Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2-3):243-278
- Feldman DP, Crutchfield J (1998) A survey of complexity measures, vol 11. Santa Fe Institute, USA
- Kadmon J, Sompolinsky H (2015) Transition to chaos in random neuronal networks. Phys Rev X 5(4):041030
- Kantz H, Schreiber T (1997) Nonlinear time series analysis. Cambridge University Press, Cambridge
- Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Global Optim 39(3):459–471
- Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, pp 1942–1948
- Koebbe M, Mayer-Kress G, Zbilut J (1994) Use of recurrence plots in the analysis of time-series data. In: Proceedings SFI studies in the science of complexity
- Kraemer KH, Donner RV, Heitzig J, Marwan N (2018) Recurrence threshold selection for obtaining robust recurrence characteristics in different embedding dimensions. Chaos 28(8):085720
- Langton CG (1990) Computation at the edge of chaos: phase transitions and emergent computation. Phys D: Nonlinear Phenom 42(1–3):12–37
- Lempel A, Ziv J (1976) On the complexity of finite sequences. IEEE Trans Inf Theory 22(1):75–81
- Marwan N, Kurths J, Saparin P (2007a) Generalised recurrence plot analysis for spatial data. Phys Lett A 360(4):545–551

- Marwan N, Romano MC, Thiel M, Kurths J (2007b) Recurrence plots for the analysis of complex systems. Phys Rep 438(5):237–329
- Marwan N, Foerster S, Kurths J (2015) Analysing spatially extended high-dimensional dynamics by recurrence plots. Phys Lett A 379:894–900
- Matsumoto M, Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans Model Comput Simul (TOMACS) 8(1):3–30
- Mitchell M, Hraber P, Crutchfield JP (1993) Revisiting the edge of chaos: evolving cellular automata to perform computations. arXiv preprint arXiv:adap-org/9303003
- Ninagawa S, Adamatzky A (2014) Classifying elementary cellular automata using compressibility, diversity and sensitivity measures. Int J Mod Phys C 25(03):1350098
- Ohira T, Sawatari R (1998) Phase transition in a computer network traffic model. Phys Rev E 58(1):193
- Orlov YL, Potapov VN (2004) Complexity: an internet resource for analysis of DNA sequence complexity. Nucleic Acids Res 32(suppl 2):W628–W633
- Packard NH, Crutchfield JP, Farmer JD, Shaw RS (1980) Geometry from a time series. Phys Rev Lett 45(9):712
- Redeker M, Adamatzky A, Martínez GJ (2013) Expressiveness of elementary cellular automata. Int J Mod Phys C 24(03):1350010
- Schinkel S, Dimigen O, Marwan N (2008) Selection of recurrence threshold for signal detection. Eur Phys J Spec Top 164(1):45–53
- Schut MC (2010) On model design for simulation of collective intelligence. Inf Sci 180(1):132–155
- Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173
- Stewart I (2000) Mathematics: the lorenz attractor exists. Nature 406(6799):948
- Takens F (1981) Detecting strange attractors in turbulence. In: Rand D, Young LS (eds) Dynamical systems and turbulence, Warwick 1980. Springer, Berlin, pp 366–381
- Tereshko V (2000) Reaction-diffusion model of a honeybee colony's foraging behaviour. In: International conference on parallel problem solving from nature. Springer, pp. 807–816
- Tereshko V, Lee T (2002) How information-mapping patterns determine foraging behaviour of a honey bee colony. Open Syst Inf Dyn 9(02):181–193
- Tereshko V, Loengarov A (2005) Collective decision making in honey-bee foraging dynamics. Comput Inf Syst 9(3):1
- Tomaszek L, Zelinka I (2016) On performance improvement of the soma swarm based algorithm and its complex network duality. In: IEEE congress on evolutionary computation (CEC) (2016). IEEE 2016:4494–4500
- Vantuch T, Zelinka I, Adamatzky A, Marwan N (2018) Phase transitions in swarm optimization algorithms. In: International conference on unconventional computation and natural computation. Springer, pp 204–216
- Wright AH, Agapie A (2001) Cyclic and chaotic behavior in genetic algorithms. In: Proceedings of the 3rd annual conference on genetic and evolutionary computation. Morgan Kaufmann Publishers Inc., pp 718–724
- Yang X-S (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, New York
- Yang X-S, M N-I (2008) Algorithms. Luniver press, Beckington, pp 242–246
- Zbilut JP, Webber CL (1992) Embeddings and delays as derived from quantification of recurrence plots. Phys Lett A 171(3–4):199–203
- Zbilut JP, Zaldivar-Comenges J-M, Strozzi F (2002) Recurrence quantification based liapunov exponents for monitoring divergence in experimental data. Phys Lett A 297(3):173–181

- Zelinka I (2004) Soma-self-organizing migrating algorithm. In: New optimization techniques in engineering. Springer, pp 167–217
- Zelinka I, Tomaszek L, Vasant P, Dao TT, Hoang DV (2017) A novel approach on evolutionary dynamics analysis-a progress report. J Comput Sci 25:437–445
- Zelinka I, Lampinen J, Senkerik R, Pluhacek M (2018) Investigation on evolutionary algorithms powered by nonrandom processes. Soft Comput 22(6):1791–1801
- Zenil H, Gauvrit N (2017) Algorithmic cognition and the computational nature of the mind. In: Encyclopedia of complexity and systems science, pp 1–9
- Ziv J, Lempel A (1978) Compression of individual sequences via variable-rate coding. IEEE Trans Inf Theory 24(5):530–536
- Zozor S, Ravier P, Buttelli O (2005) On lempel-ziv complexity for multidimensional data analysis. Phys A Stat Mech Appl 345(1):285-302

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.