Expressive tools of meta-heuristic population algorithms for continuous global optimization

Anatoly Karpenko[a], Inna Kuzmina

[a] Bauman Moscow state technical university, 2-nd Baumanskaya, 5, Moscow, 105005, Russian Federation

Abstract

Meta-heuristic population algorithms for continuous global optimization, inspired by collective processes in living and inanimate nature, in human society, are considered. The method of the expressive means these algorithms formalization is proposed. It allows to create their patterns –concise and easy-to-read formalized descriptions of the algorithm structure and main features.

Keywords [1]

Global optimization, meta-heuristic algorithm, population algorithm, algorithm pattern

1. Introduction

In various publications, meta-heuristic population algorithms for global search optimization (MEP-algorithms) are called behavioral, intellectual, inspired by nature and human society, swarming, multi-agent, etc. [1]. Population algorithms are based on the concept of a population – a finite set of candidates for a solution. The population consists of individuals and evolves during generation changes (i.e. during algorithm iterations). At each iteration, all individuals in the population move, possibly excluding only some of them (for example, the "best" individuals).

Modern population algorithms are meta-heuristic and stochastic [2]. There are numerous examples of successful solutions to complex practical problems of global optimization by using MEP-algorithms, for example, computer-aided design problems. MEP-algorithms are numerous and very varied, so the book [2] introduced more than 130 such algorithms, not counting their numerous modifications. The diversity of MEP-algorithms creates a problem of classification and formalization of their expression means. This causes the problem of developing concise and easy-to-read formalized descriptions of these algorithms structure and main features, called patterns.

To solve this problem, a classification and formalization of the MEP-algorithms entities and the evolutionary operators are proposed. Rules for developing MEP-algorithm patterns using formal descriptions of these entities and operators are presented. An example of a the widely known MEP-algorithm PSO (Particle Swarm Optimization) pattern is given [3].

2. Problem statement, MEP-algorithms basic designations and classification

Deterministic static problems of global conditional and unconditional minimization are considered

$$\min_{X \in R^{|X|}} f(X) = f(X^*) = f^*,$$

$$\min_{X \in \Omega \subset R^{|X|}} f(X) = f(X^*) = f^*,$$

[1] Russian Advances in Fuzzy Systems and Soft Computing: selected contributions to the 8-th International Conference on Fuzzy Systems, Soft Computing and Intelligent Technologies (FSSCIT-2020), June 29 – July 1, 2020, Smolensk, Russia

EMAIL: apkarpenko@mail.ru

where $|X|$ – is the vector of variable parameters X dimension; $f(X)$ – is an object function with values in space $R^{|X|}$; X^*, f^* – the desired optimal solution and the object function value, respectively; Ω – a convex set of acceptable vector X values defined by the set of limiting functions $\Gamma(X)$.

The approach to the problem solution (the candidate for the solution) is called an *individual* and denoted $s_i, i \in [1:|S|]$. Here S – is a set of individuals called a *population*; $|S|$ – population capacity (number of individuals).

The *state* of an individual is usually determined by the vectors of its position X_i, speed ΔX_i, and acceleration $\Delta^2 X_i$, so the state vector of this individual is equal to

$$V_i = X_i \cup \Delta X_i \cup \Delta^2 X_i$$

The *fitness* of an individual $s_i \in S$ in its current position determined by the fitness function value $\varphi_i = \varphi(f(X_i)) = \varphi(X_i)$. The fitness is the most important characteristic of the individual. It is an personal characteristic of the individual current quality (in other words, its "self-esteem"). It is considered that the fitness function $\varphi(X)$ is also subject to minimization.

The general scheme of the MEP-algorithm has the following form.

1. Algorithm initialization. Sets the initial value of the iterations number counter $t = 0$, the initial states of individuals $V_i(0), i \in [1:|S|]$, i.e. the initial population is generated, and the values of the free parameters of the algorithm are also set.

2. The evolutionary operators of this algorithm are executed, own for the each vector $V_i = V_i(t)$ components. As a result, the individual get a new position in the search area $X_i(t+1) = X'_i$, a new speed $\Delta X_i(t+1) = \Delta X'_i$ and a new acceleration $\Delta^2 X_i(t+1) = \Delta^2 X'_i$, $i \in [1:|S|]$.

3. Checks whether the conditions for completing the evolutionary process are met. If these conditions are not met, $t = t + 1$ is assumed and the transition to the step 2 is implemented. Otherwise, the best of the found positions of individuals is taken as an approximate solution to the problem.

From a fairly general point of view, the evolution of an individual $s_i \in S$ in the course of iterations is determined by the function $\mathrm{E}(\cdot)$ (evolutionary operator) which has the following type

$$X_i(t+1) = \mathrm{E}\left(X_j(\tau), \varphi(X_j(\tau)), \Gamma(X_j(\tau)), j \in [1:|S|], \tau \in [1:t]\right).$$

The sense of this formula is that the new position of an individual $s_i \in S$ determines, in general, all previous positions of all individuals in the population, including current individual, as well as the corresponding values of fitness and limiting functions.

Most MEP-algorithms are iterative, so the function $\mathrm{E}(\cdot)$ remains the same for all iterations, excluding perhaps only a small number of initial "overclocking" iterations.

Generally speaking, MEP-algorithms can be classified by a large number of features, the main ones being the following [2]:

- single-and multi-population algorithms;
- algorithms with and without adaptation, self-adaptation algorithms;
- stationary and non-stationary algorithms;
- synchronous and asynchronous algorithms;
- uniform and non-uniform algorithms;
- single-step and multi-step algorithms;
- non-sequential and trace algorithms;
- non-composite (simple) and composite (hybrid) algorithms;
- single-stage and multi-stage algorithms.

Among the evolutionary MEP-algorithms, there are still algorithms based on different models of evolution (Darwin, Lamarck, Baldwin, Etc.) [4].

3. MEP-algorithms entities and their specifications

For a formalized description of MEP-algorithm entities, the following descriptors are used:

$$system \in \{det, stoch, chaotic\};$$
$$control \in \{stat, prog, adapt, prog-adapt, self-adapt\};$$
ID.

The descriptor *system* determines whether an entity is deterministic, stochastic, or chaotic, respectively. The *control* descriptor identifies static, program-changing, adaptive, program-adaptive, and self-adaptive entities, respectively. *ID* details the nature of an entity's dependency on the values of its arguments. The set of possible values for the system and control descriptors for some entities can be reduced. The concept of adaptation and self-adaptation mechanisms is understood in accordance with [5, 6].

One or more characteristics of this entity can be associated with each of the MEP- algorithm entities. Static entities are entities that have static characteristics. The characteristics of dynamic entities are dynamic.

One of the important MEP-algorithms features is that they have *free* parameters, the values of which are set by the Decision Maker (DM) based on the preferences and/or features of the optimization problem. Meta parameters are auxiliary free parameters that define the values of the main free parameters. Meta parameters are always deterministic static user-defined parameters. The General format of the parameter set specification is

$$\text{parameters}\Big|_{ID}^{system.control} P : \text{<conditions>. -- <comments>}$$

where for meta-parameters ID $= meta$, and for free parameters – ID $= free$.

Next, for writing ease, omit *conditions* and *comments* in the specifications.

- General specification formats for *individuals* and *populations*:

$$\text{individual}\Big|_{ID}^{system.control} \{s, V, \varphi(V_i)\}.$$

$$\text{population}\Big|_{ID}^{system.control} \{s_i; i \in [1:|S|]\} = S.$$

- The format specification of an entity Ξ *characteristics*:

$$\text{characteristic}\Big|_{ID}^{system.control} Name(\Xi, A).$$

Here *Name* – is the name of the entity; *A* – is a set of arguments that may include free parameters of the algorithm.

Example. Population characteristics used by the Evolution Strategy (ES) algorithm [7]:

$$\text{characteristic}\Big|_{ES}^{det.adapt} \varpi(S, \Delta t). \text{ -- Percentage of successful mutations}$$

This refers to mutations produced by the corresponding evolutionary operator of the ES algorithm during iterations $[(t-\Delta t):t]$, where Δt – is free parameter of the algorithm. The characteristic is deterministic and adaptive. The ES algorithm determines the rules for calculating characteristic values.

- The following *neighborhood spaces* are predefined: R_φ – target space $\Re^1$; R_X – search space $\Re^{|X|}$; R_T – "topological" space. The elements of these spaces are (respectively): the values of the fitness function $\varphi(X)$; the components of the variable parameters vector X; individuals $s_i, i \in [1:|S|]$. Space R_φ is one-dimensional continuous, R_X – is $|X|$-dimensional continuous, and R_T – is a discrete dimension of $|S|$.

Various *metrics of individuals proximity* can be defined in listed spaces. The general format of metrics is as follows:

$$\text{metric}\Big|_{R.ID}^{system.control} \mu(s_i, s_j, A).$$

where $\{s_i, s_j\} \in S$; $R \in \{R_\varphi, R_X, R_T\}$.

For the specification of a topological space R_T, a *neighborhood graph* must be defined. It specified as

$$\text{graph}\Big|_{R.ID}^{system.control} NG(\Sigma, A) .$$

Here $\Sigma \subseteq S$ – is the considered combination of individuals (population, subpopulation, etc.).

Example. The *Particle Swarm Optimization* (PSO) algorithm can use a *ring*-type neighborhood graph. The specification of this graph has the form:

$$\text{graph}\Big|_{R_t.ring}^{det.stat} NG(S).$$

This specification defines a deterministic static neighborhood graph of all individuals in population S in a topological space R_T.

- The specification of the *area* D_i of an individual $s_i \in S$ in space $R \in \{R_\varphi, R_X, R_T\}$ has the form:

$$\text{area}\Big|_{R.ID}^{system.control} D_i(s_i, A) .$$

Here, the arguments A necessarily include a parameter or parameters that determine the size of the area and its position relative to the current location of the individual s_i (that is, relative to the point X_i). In different MEP-algorithms, the area of an individual is called differently, for example, in the ant algorithm COAK, the surrounding area is called the search region [8].

- A population (subpopulation) may include a number of individuals *unions*. The specification of union U in space R has the form

$$\text{union}\Big|_{R.ID}^{system.control} U(S, A) .$$

The uniion $U(S, A)$ can be constructed, in particular, based on a area $D_i(s_i, A)$ by a rule:

$$U_i(S, D_i) = \{s_{i_j} \in D_i(s_i, A)\} \subseteq S .$$

- An individual s_i in the process of its evolution over the course of generations $[0:t]$ consistently occupies positions $X_i(0)$, $X_i(1)$, … , $X_i(t)$ in the search space R_X. The set of these positions produce the *track* of the individual s_i:

$$\text{track}\Big|_{R_X.ID}^{system.control} Tr_i(s_i, 0, t, A) .$$

- Some MEP-algorithms, primarily ant algorithms, use *pheromone tracks*, which are specified as

$$\Phi\text{-track}\Big|_{R_X.ID}^{system.control} \Phi_i(s_i, t^\phi, A) .$$

Here $t^\phi \in [0:t]$ – the iteration number when the individual s_i left a pheromone track. The specification defines a set $\left\{\{X_i(\tau), \phi_i(\tau)\}; \tau \in [1:t]\right\}$, where $\phi_i(\tau)$ – is the current intensity of the pheromone label at the point $X_i(\tau)$. The change in pheromone concentration over time is determined by the operator renovation (see below).

- MEP-algorithms widely use *attractors* – points in the search space, generally speaking, that do not coincide with the position of any individual or the point of its track, which determine the direction of this individual or individuals movement. In the general case, the specification of the attractor has the form:

$$\text{attractor}\Big|_{to/from.ID}^{system.control} O(\Xi, A) .$$

where Ξ – the entity that the attractor O is based on. There are attractive, repulsive, attractive/repulsive attractors (to, from, to/from, respectively).

Example. The attractor used by the *Monkey Algorithm* (*MA*) and many other MEP-algorithms [9] can be specified as

$$\text{attractor}\Big|_{to.\,CG}^{det.\,adapt}\ X^c(S).$$

The specification defines the attracting attractor $X^c \in R_X$ of the population S, which has the meaning of the current coordinates $|X|$-dimensional vector – the *Center of Gravity* (*CG*) – of all $|S|$ individuals in this population.

- MEP-algorithms can use a number of other entities: *memory*; *tunnel* between two individuals or their aggregates; *event* in a population or association of individuals; *channel* for exchanging information between individuals or their aggregates; *message*, etc.

4. MEP-algorithms operators and their specifications

The main groups of operators are as follows.

- Initialization and completion of the algorithm:

$$\text{init}\Big|_{ID}^{system.\,control}\ \Xi(A);$$
$$\text{stop}\Big|_{ID}^{system.\,control}\ \Sigma(t).$$

Here, the first operator initializes the entity Ξ, the second operator initializes the completion of the individuals combination Σ evolution process.

- Creating, calculating and updating entity attributes. Operators that implement these actions with respect to an entity Ξ have the same format:

$$\text{creation (calculation, renovation)}\Big|_{ID}^{system.\,control}\ \Xi(A).$$

- Evolution (movement) of the individuals in the search space R_X. The operator is the main evolutionary operator of MEP-algorithms. The first of the following specifications

$$\text{motion}\Big|_{n.\,ID}^{system.\,control}\ \Sigma \to \Sigma'(A);$$
$$\text{try\_motion}\Big|_{n.\,ID}^{system.\,control}\ \Sigma \to \Sigma'(A);$$

means that the combination $\Sigma = \Sigma(\mathrm{t})$ will be transformed into the combination $\Sigma' = \Sigma(\mathrm{t}+1)$ as a result of executing the operator. Each individual in the combination Σ is moved using information about other n individuals in the combination.

MEP-algorithms often use virtual (trial) movements of individuals. These movements are defined by the second of the presented above specifications.

- Local and global search. Often, MEP-algorithms are hybridizations based on the use of known or original local optimization algorithms [1]. Also, MEP-algorithms often include original global optimization algorithms. The specifications below describe these algorithms in relation to the combination of individuals Σ.

$$\text{local\_search}\Big|_{ID}^{system.\,control}\ \Sigma \to \Sigma'(A).$$
$$\text{global\_search}\Big|_{ID}^{system.\,control}\ \Sigma \to \Sigma'(A).$$

- Managing combinations of individuals. In MEP-algorithms, a large group is formed by operators for managing populations of individuals. Examples of operator specifications for this group are provided below. The meaning of operators follows from their names and comments.

$$\text{association}\Big|_{n.\,ID}^{system.\,control}\ \{\Sigma_1, \Sigma_2, ..., \Sigma_n\} \to \Sigma(A) : \Sigma = \bigcup\ \{\Sigma_1, \Sigma_2, ...\}.$$
$$\text{deleting}\Big|_{n.\,ID}^{system.\,control}\ \Sigma \to \Sigma'(s_{i_1}, ..., s_{i_n}) :\ |\Sigma'| \le |\Sigma|;\ |\Sigma'| = |\Sigma| - n.$$

$$\text{expansion}\Big|_{n.ID}^{system.control}\ \Sigma \to \Sigma'(s_{i_1},...,s_{i_n}):\ |\Sigma'| \ge |\Sigma| + n.$$

$$\text{compression}\Big|_{n.ID}^{system.control}\ \Sigma \to \Sigma'(A):\ |\Sigma'| \le |\Sigma|;\ |\Sigma'| = n.$$

$$\text{ordering}\Big|_{ID}^{system.control}\ \Sigma \to \Sigma'(A):\ |\Sigma'| = |\Sigma|.$$

$$\text{shake-up}\Big|_{ID}^{system.control}\ \Sigma \to \Sigma'(A):\ |\Sigma'| = |\Sigma|.$$

$$\text{selection}\Big|_{n.ID}^{system.control}\ \Sigma \to \Sigma'(A):\ \Sigma' \subset \Sigma;\ |\Sigma'| = n.$$

$$\text{splitting}\Big|_{n.ID}^{system.control}\ \Sigma \to \{\Sigma_1(A),\Sigma_2(A),...,\Sigma_n(A)\}: \Sigma = \bigcup\ \{\Sigma_1,\Sigma_2,...,\Sigma_n\}.$$

$$\text{replacement}\Big|_{ID}^{system.control}\ \{\Sigma_1,\Sigma_2\} \to \{\Sigma_1'(A),\Sigma_2'(A)\}:\ \Sigma_1' = \Sigma_2, \Sigma_2' = \Sigma_1.$$

$$\text{replication}\Big|_{n.ID}^{system.control}\ \Sigma \to \Sigma'(A):\ |\Sigma'| = \{\Sigma;\Sigma;...,\Sigma\};\ |\Sigma'| = n|\Sigma|.$$

- Other operators can be used in the patterns of MEP-algorithms, including the following:

$$\text{comeback}\Big|_{\Omega.ID}^{system.control}\ \Sigma \to \Sigma'(\Omega):\ |\Sigma'| = |\Sigma|.$$ -- Ω - search area

$$\text{send}\Big|_{ID}^{system.control}\ \{X_i,\varphi(X_i)\} \to s_j.$$

$$\text{receive}\Big|_{ID}^{system.control}\ \{X_j,\varphi(X_j)\} \to M_i.$$ -- M_i - "memory" of an individual s_i

Operator *comeback* returns individuals of the combination $\boldsymbol{\Sigma}$ to a valid area $\boldsymbol{\Omega}$ if they go beyond the boundaries of this area. Operator *send* causes the sending of a message $\{\boldsymbol{X_i}, \boldsymbol{\varphi}(\boldsymbol{X_i})\}$ by individual $\boldsymbol{s_i}$ to other individual $\boldsymbol{s_j}$, and the operator *receive* – receiving by individual $\boldsymbol{s_i}$ the same messages from individual $\boldsymbol{s_j}$. The message may also contain other information.

5. Example. Particle Swarm Optimization (PSO) algorithm and its pattern [3]

The problem of multidimensional global unconditional minimization in the initial parallelepiped $\Pi = (X^-;\ X^+)$ is considered.

5.1. Algorithm

1. *Initialization*

1.1. User initialization of free parameters $P_{free} = \{|S|, |X|, X^-, X^+, \Delta X^-, \Delta X^+, b_I, b_C, b_S\}$. Recommended value: $b_I = 0{,}7298$, $b_C = b_S = 1{,}49618$.

1.2. *Population initialization*. Evenly randomly distributed individuals of the population in the region Π, and the vectors of their initial "speeds" $\Delta X_i, i \in [1{:}\,|S|]$ in the area $\Pi_\Delta = [\Delta X | \Delta X^- \le \Delta X \le \Delta X^+]$ where the inequalities are understood piecemeal.

2. Population evolution

The scheme of the individual $s_i, i \in [1{:}\,|S|]$ evolution algorithm has the following form.

2.1.In the track of the individual s_i, the best point X_i^{hbest} is chosen such that

$$\min_{\tau \in [0:t]} \varphi(X_i(\tau)) = \varphi\left(X_i^{hbest}\right).$$

2.2. According to the neighborhood topology used for individuals in the population, the best individual $s_{i^*}^{NBbest}$ is selected from the nearest current neighbors $NB_i \subseteq S$ of an individual s_i:

$$\min_{s_j \in NB_i} \varphi\left(X_j\right) = \varphi\left(X_{i^*}^{NBbest}\right).$$

2.3. A randomizing three step operator for moving an individual s_i is used:

$$X'_i = X_i + \Delta X_i; \quad (1)$$

$$\Delta X_i = b_I \Delta X_i^- + U_{|X|}(0; b_C) \otimes \left(X_i^{hbest} - X_i\right) + + U_{|X|}(0; b_C) \otimes \left(X_{i^*}^{hbest} - X_i\right). \quad (2)$$

Here $\Delta X_i^- = \Delta X_i(t-1)$; $U_{|X|}(a;b)$ – $|X|$-dimensional vector of random numbers evenly distributed in the interval $[a, b]$; $\otimes$ – symbol of the direct multiplication of vectors.

3. *The evolutionary process completion*
Search end conditions are not fixed.

5.2. Algorithm pattern

Algorithm Particle Swarm Optimization, PSO
specifications

parameters $\left|{}^{det.stat}_{free}\right. \{|S|, |X|, X^-, X^+, \Delta X^-, \Delta X^+, b_I, b_C, b_S\} = P_{free}$: $b_I = 0{,}7298$, $b_C = b_S =$ 1,49618.

individual $\left|{}^{det.stat}\right. \{X, \Delta X, \varphi(X)\} = s$.

population $\left|{}^{det.stat}\right. \{s_i; i \in [1:|S|]\} = S$.

graph $\left|{}^{det.stat}_{R_T}\right. NG(S)$. -- graph of the individuals neighborhood in the space R_T

$\left|{}^{|S|}_{i=1}\right|$ union $\left|{}^{det.stat}_{R_T}\right. NB_i(s_i, NG)$. -- neighbors of the individual s_i

$\left|{}^{|S|}_{i=1}\right|$ track $\left|{}^{det.stat}_{R_T}\right. Tr_i(s_i, 0, t)$. -- trace of an individual s_i in space R_X

initialization

init $\left|{}^{det.stat}_{DM}\right. P_{free}$.

init $\left|{}^{stoch.prog}_{uniform}\right. S(X^-, X^+, \Delta X^-, \Delta X^+) = S(0)$.

$\left|{}^{|S|}_{i=1}\right|$ creation t $\left|{}^{det.stat}_{DM}\right. NB_i(s_i, NG)$.

evolution

$\left|{}^{|S|}_{i=1}\right|$ selection t $\left|{}^{det.stat}_{1.\varphi\_\ best}\right. Tr_i(s_i, 0, t) \to X_i^{hbest}$. -- the best position of an individual s_i in its track

$\left|{}^{|S|}_{i=1}\right|$ selection $\left|{}^{det.stat}_{1.\varphi\_\ best}\right. NB_i \to s_{i^*}^{NBbest}(NG)$. -- the best individual from among the neighbors of the individual s_i

$\left|{}^{|S|}_{i=1}\right|$ $\begin{cases} \text{motion } \left|{}^{stoch.stat}_{3.uniform}\right. \Delta X_i \to \Delta X_i'(\Delta X_i^-, X_i^{hbest}, X_{i^*}^{NBbest}, b_I, b_C, b_S). \\ \text{motion } \left|{}^{det.stat}_{1\times 1}\right. X_i \to X_i' \ (\Delta X_i') = X_i(t+1). \end{cases}$

termination

stop $\left|{}^{any}\right. S(t+1)$.

5.3. Conclusion~~Discussion~~

The presented pattern allows, first, to get the most general information: the algorithm is a track, single-population, not hybrid. Second, the pattern allows to get the following detailed information about this algorithm.

- The *specifications* section. The algorithm has a small number of deterministic static free parameters. The vector of state variables of individuals is composite: $V = (X, \Delta X)$. The algorithm uses the neighborhood of individuals in space R_T. The neighborhood topology is defined by the neighborhood graph $NG(S)$, so that the neighbors of an individual s_i are the individuals of the association $NB_i(s_i, NG)$. Traces (not pheromone) of all individuals in the population have the form $Tr_i(s_i, 0, t)$.
- The *initialization* section. The algorithm uses the usual initialization procedures for any MEP-algorithm for its free parameters and initial states of individuals in the population. In addition, at the initialization stage, the DM creates neighborhood graphs $NB_i(s_i, NG)$ for each individual $s_i, i \in [1:|S|]$. Free parameters initialization is performed by the DM, initialization of the initial states of individuals in the population is performed programmatically.
- The *evolution* section. At the beginning of each iteration, the PSO-algorithm determines the "historically" best position X_i^{hbest} of the individual s_i in each tracks $Tr_i(s_i, 0, t), i \in [1:|S|]$. Then, for each individual s_i $i \in [1:|S|]$, the algorithm finds the best neighboring individual $S_{i^*}^{NBbest}$, and thus a point $X_{i^*}^{NBbest}$ in the search space. The last group of the pattern consisting of two operators implements the movement of each individual in the spaces R_X and $R_{\Delta X}$ by formulas (1), (2).
- The *termination* section. The *stop* operator specifier shows that the PSO-algorithm does not fix the iterative optimization process end criteria.

6. Acknowledgements

Systematization and formalization of the MEP-algorithms expressive means allows to put in practice flatness the problem of automated structural-parametric synthesis of algorithms that are in some sense the best for a given class of global optimization problems.

7. References

[1] A. P. Karpenko, Sovremennye algoritmy poiskovoj optimizacii. Algoritmy vdohnovlennye prirodoj. M.: Izdatel'stvo MGTU im. N. E. Baumana, 2014.
[2] Bo Xing, Wen-Jing Gao, Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms. – Springer International Publishing Switzerland, 2014.
[3] M. E. H. Pedersen, Chipperfield A. J. Simplifying Particle Swarm Optimization / Applied Soft Computing, 2010, Vol. 10, Issue 2, March, 2010, 618-628.
[4] Evolutionary Computation. 1. Basic Algorithms and Operators / Edited by Thomas Back, David B. Fogel and Zbignew Michalevich. – Institute of Physics Publishing, Bristol and Philadelphia, 2004.
[5] A. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 3(2) (1999) 124-141.
[6] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing. Second Edition, Natural Computing. Springer-Verlag Berlin Heidelberg, 2015.
[7] Hong-Kyu Kim, Jin-Kyo Chong, Kyong-Yop Park, David A. Lowther, Differential Evolution Strategy for Constrained Global Optimization and Application to Practical Engineering Problems, IEEE Transactions on Magnetics, 43, 4 (2007). 1565-1568.
[8] Serap Ulusam Seçkiner, Yunus Eroğlu, Merve Emrullah, Türkay Dereli. Ant colony optimization for continuous functions by using novel pheromone updating, Applied Mathematics and Computation, 219, 9 (2013) 4163-4175.
[9] Ruiqing Zhao, Wansheng Tang, Monkey Algorithm for Global Numerical Optimization, Journal of Uncertain Systems, 2, 3 (2008) 165-176.