



Kirill V. Pushkaryov

## Global optimization via neural network approximation of inverse coordinate mappings with evolutionary parameter control

**ABSTRACT.** A hybrid method of global optimization NNAICM-PSO is presented. It uses neural network approximation of inverse mappings of objective function values to coordinates combined with particle swarm optimization to find the global minimum of a continuous objective function of multiple variables with bound constraints. The objective function is viewed as a black box.

The method employs groups of moving probe points attracted by goals like in particle swarm optimization. One of the possible goals is determined via mapping of decreased objective function values to coordinates by modified Dual Generalized Regression Neural Networks constructed from probe points.

The parameters of the search are controlled by an evolutionary algorithm. The algorithm forms a population of evolving rules each containing a tuple of parameter values. There are two measures of fitness: short-term (charm) and long-term (merit). Charm is used to select rules for reproduction and application. Merit determines survival of an individual. This two-fold system preserves potentially useful individuals from extinction due to short-term situation changes.

Test problems of 100 variables were solved. The results indicate that evolutionary control is better than random variation of parameters for NNAICM-PSO. With some problems, when rule bases are reused, error progressively decreases in subsequent runs, which means that the method adapts to the problem.

*Key words and phrases:* global optimization, heuristic methods, evolutionary algorithms, neural networks, parameter setting, parameter control, particle swarm optimization.

2010 *Mathematics Subject Classification:* 90C26; 90C59,

## Introduction

Global optimization of multivariate objective functions (OFs) viewed as black boxes with minimal restrictions on the function's properties is an important problem because it promises a universal way to solve many practical problems of engineering, machine learning, etc.

Nowadays, nature-inspired computing is actively developing. It solves computational problems through imitation of natural phenomena. In optimization this approach gave birth to evolutionary algorithms (evolutionary programming, evolutionary strategies, genetic algorithms, differential evolution) [1, 2] and a lot of other optimization algorithms based on various physical, chemical and biological phenomena [3], such as simulated annealing and particle swarm optimization (PSO).

Although for absolutely unrestricted OFs efficiency of all methods averaged over all possible problems is equivalent [4], methods that work with more restricted practically possible problems are of great interest. These methods must make the best use of the information they obtain by evaluating an OF at probe points. Long-term preservation of information relevant for global minimum search is an important step towards this goal.

Proper parameter setting is crucial for optimization algorithm efficiency. Optimal choice of parameters is an optimization problem in itself, where some measure of quality dependent on parameters must be optimized. Parameters may be set manually or automatically before the start of an algorithm (the parameter tuning problem) or during its execution (the parameter control problem) [5].

Manual parameter setting requires a good understanding of how parameters affect an algorithm in various situations, which in turn entails expert knowledge and experience. The data on effects and relations of parameters gathered for one problem may be inapplicable to another. Experimental exploration of parameters is difficult because the problem at hand must be solved multiple times with different parameters, so resource requirements increase manifold.

The optimal values of parameters may be different not only for different problems but also for different stages of a search [6, p. 1]. For example, gradual reduction of the inertia weight or increasing of the neighbourhood size was suggested for PSO [7, pp. 36, 40]. A parameter control mechanism can actively interact with the search process and learn. As a result, the problem of parameter control is particularly important.

The problem of automatic parameter setting attracted a lot of attention in evolutionary optimization. A review of various approaches is presented

in [5, 6]. The range of methods is diverse: from random variation and deterministic heuristics to controllers based on fuzzy logic [8, 9], predefined heuristic rules [10], reinforcement learning [11–13] or an auxiliary meta-evolutionary algorithm [12]. In [5] parameter setting methods are classified into deterministic (no feedback from the algorithm being configured, parameters are set according to a predetermined schedule), adaptive (feedback from the algorithm is taken into account), self-adaptive (parameter values are encoded in the genome along with the solutions and evolve together).

Neural network approximation of inverse coordinate mappings (NNAICM) is a heuristic method for finding the global minimum of a continuous multivariate “black box” objective function with simple bounds on the variables. The method is described in detail in [14, 15]. It was included in the hybrid heuristic parallel method [16], where fixed heuristics without learning were employed to control its parameters.

Here we present a hybrid global optimization method based on NNAICM and PSO with evolutionary parameter control. NNAICM helps to find the goal to which a point (a particle in PSO terms) moving by the rules of PSO is attracted. An evolutionary algorithm controls the parameters of NNAICM at the same time.

The parameter control task in this work is special in that the parameters pertain to individual agents (particles), not to the algorithm as a whole. Hence, at each moment there may be multiple optimal parameter sets for different agents.

A numerical global optimization problem is stated as follows. Consider an objective function  $\Phi(\mathbf{x})$  defined on a bounded set  $\Omega$ , where

$$(1) \quad \Omega = \{\mathbf{x}: L_i \leq x_i \leq U_i, i = \overline{1, D}\} \subset \mathbb{R}^D.$$

Find an approximate minimum of the function  $\Phi_{min}^*$  and a point  $\mathbf{x}_{min}^*$  where it’s attained:

$$(2) \quad \Phi_{min} = \min_{\mathbf{x} \in \Omega} \Phi(\mathbf{x}) = \Phi(\mathbf{x}_{min}),$$

$$(3) \quad \Phi_{min}^* = \Phi(\mathbf{x}_{min}^*) \leq \Phi_{min} + \varepsilon_{\Phi},$$

where  $\varepsilon_{\Phi} > 0$  is a tolerance that specifies the required accuracy of the solution.

## 1. Neural Network Approximation of Inverse Coordinate Mappings

NNAICM is based on iterative decreasing of OF values and mapping them to coordinates by a Generalized Regression Neural Network (GRNN).

The GRNN is trained on samples  $\langle \Phi(\mathbf{x}), \mathbf{x} \rangle$ , which consist of an OF value and the corresponding coordinates. Therefore, the GRNN is said to approximate an inverse coordinate mapping. One-step learning is an important advantage of GRNN — it may be constructed from samples in one step.

The basic NNAICM algorithm is as follows:

- (1) Initialization. Set  $k := 1$ . Fill  $P^{[k]}$  — a starting set of probe points from the search space, which may be chosen at random.
- (2) Train the network  $\text{GRNN}^{[k]}$  to map OF values to coordinates on the samples  $\{\langle \Phi(\mathbf{x}), \mathbf{x} \rangle : \mathbf{x} \in P^{[k]}\}$ .
- (3) Take the point  $\mathbf{x}_{min}^{[k]}$  with the minimum OF value from the set  $P^{[k]}$ . Lower the value by some *decrement*  $\varepsilon_f^{[k]}$  and map it to coordinates by the GRNN:  $\mathbf{x}_{min}^{*[k]} = \text{GRNN}^{[k]}(\Phi(\mathbf{x}_{min}^{[k]}) - \varepsilon_f^{[k]}, s_\varphi)$ , where  $s_\varphi$  is an approximation smoothing parameter.
- (4) The new point is included in the set  $P^{[k+1]}$  if its OF value is less than the current minimum:

$$(4) \quad P^{[k+1]} = \begin{cases} P^{[k]} \cup R^{[k]} \cup \{\mathbf{x}_{min}^{*[k]}\} & \text{if } \Phi(\mathbf{x}_{min}^{*[k]}) < \Phi(\mathbf{x}_{min}^{[k]}), \\ P^{[k]} \cup R^{[k]} & \text{otherwise,} \end{cases}$$

where  $R^{[k]}$  is a set of random probe points.

- (5)  $k := k + 1$ .

- (6) Repeat steps 2–5 until stop conditions are satisfied.

These steps for the function  $y = x^2$  are shown in Figure 1.

In Figure 1 we made a GRNN approximation of an inverse coordinate mapping based on probe points  $P_1$ – $P_5$ . Then we used it to obtain a new point  $P_6$  and refined the approximation taking  $P_6$  into account.

Practical application of the method described above meets with some difficulties. Its efficiency quickly diminishes when the number of variables goes up. It can get stuck because each iteration starts with a single (best) probe point and random generation is not an effective source of new probe points. To conquer this problem, NNAICM was included in the hybrid heuristic parallel method [16] in which multiple optimization agents search for the global minimum. The agents iteratively execute a set of optimization algorithms (search tools), which share a common set of probe points.

When an OF has a number of minima close by value, the inverse mapping line can miss all of them by far. To overcome this issue, a special

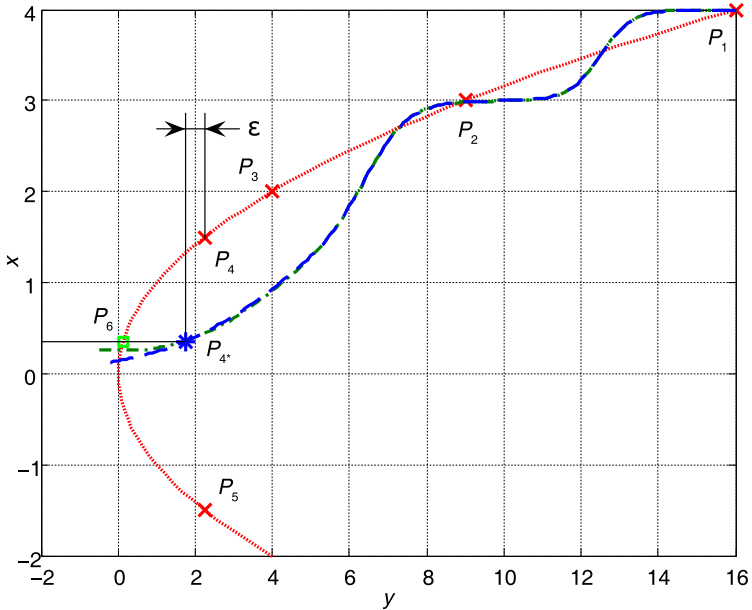


FIGURE 1. Principle of the basic NNAICM algorithm

modification of GRNN was proposed — Dual GRNN (DGRNN) [17]. Besides the input  $\varphi$  for OF values it has an additional input  $\mathbf{x}_f$  taking coordinates of a point — a *search focus*. A training sample of DGRNN consists of an input *exemplar*  $\varphi_i$  and an output exemplar  $\mathbf{X}_i$ . The farther is the memorized exemplar from the focus, the lower is its activation and thus its contribution to the output, so exemplars near the focus have an advantage. DGRNN has an additional smoothing parameter  $s_x$ , which is an analog of  $s_\varphi$  for the second input. The first hidden layer of DGRNN measures similarity between input values and exemplars. Similarity coefficients are multiplied for each pair of an input and an output exemplar in the second layer. The resulting coefficients are used for weighted summation of output exemplars in the third and fourth layers.

In Figure 2 the DGRNN schematic for two pairs of exemplars is shown. The connection weights not equal to 1 are printed nearby.

Finally, control of the  $\varepsilon_f$ ,  $s_\varphi$ ,  $s_x$  parameters and the search focus poses a problem. In [16] DGRNN wasn't used and fixed heuristics without learning were employed to control  $\varepsilon_f$  and  $s_\varphi$ . The decrement of the OF value was computed as  $\varepsilon_f^{[k]} = 3(\Phi_{max}^{[k]} - \Phi_{min}^{[k]})/N_p$ , where  $\Phi_{max}^{[k]}$  and  $\Phi_{min}^{[k]}$

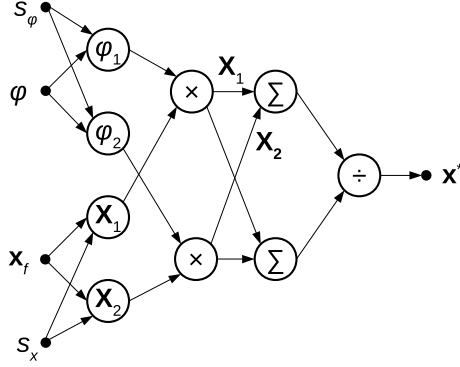


FIGURE 2. DGRNN schematic for two pairs of exemplars

are the maximum and the minimum OF values over the probe point set at the  $k$ th iteration,  $N_p$  is the number of the probe points. To simplify setting of  $s_\varphi$ , the distance function of the first GRNN layer was changed to:

$$(5) \quad \text{dist}^{[k]}(\varphi_1, \varphi_2) = \frac{|\varphi_1 - \varphi_2|}{\Phi_{max}^{[k]} - \Phi_{min}^{[k]}}.$$

This makes the distances and  $s_\varphi$  relative quantities.

## 2. Hybrid NNAICM-PSO Method with Evolutionary Parameter Control

To improve its efficiency, NNAICM was combined with PSO and evolutionary parameter control.

*Base points* (BPs) are counterparts of PSO particles in the discussed method. The number of BPs  $N_b$  is a parameter. Each BP at the  $k$ th iteration has the position  $\mathbf{b}^{[k]}$ , the velocity  $\mathbf{v}^{[k]}$  and the *private goal*  $\mathbf{g}_i^{[k]}$ . Initially,

$$(6) \quad \mathbf{b}^{[1]} = \mathbf{g}_i^{[1]} = \mathbf{U}[\mathbf{L}, \mathbf{U}],$$

$$(7) \quad \mathbf{v}^{[1]} = \mathbf{U}[-k_{v1}(\mathbf{U} - \mathbf{L}), k_{v1}(\mathbf{U} - \mathbf{L})],$$

where  $\mathbf{L} = (L_1, \dots, L_D)$ ,  $\mathbf{U} = (U_1, \dots, U_D)$  are the bounds on the search space;  $\mathbf{U}[\mathbf{l}, \mathbf{u}]$  is a vector of numbers uniformly distributed on the set  $\{\mathbf{x}: l_i \leq x_i \leq u_i, i = \overline{1, D}\}$ ;  $k_{v1}$  is a parameter.

BPs are arranged in disjoint groups by  $S_{bg}$  points. Every group has the *group goal*  $\mathbf{g}_g^{[k]}$  that is selected each iteration by the lowest OF value

from the list:  $\mathbf{g}_g^{[k-1]}$ , private goals of the group. A group goal may be improved by local search and trajectory extrapolation as explained below.

To prevent search stagnation, inactive BPs are restarted every  $I_{rest}$  iterations by random reinitialization described above. A BP is restarted at the  $k$ th iteration if all conditions are satisfied:

$$(8) \quad \|\mathbf{v}^{[k]}\| < v_{min},$$

$$(9) \quad \Phi_{min,b}(k - I_{rest}) - \Phi_{min,b}(k) < \varepsilon_b I_{rest},$$

where  $\Phi_{min,b}(k) = \min_{i=1,k} \Phi(\mathbf{b}^{[i]})$  is the minimum OF value over all positions of the BP from the first to the  $k$ th iteration;  $v_{min}$ ,  $\varepsilon_b$ ,  $I_{rest}$  are parameters of the algorithm.

A variant of DGRNN, QDGRNN (Quantile DGRNN), was created for NNAICM-PSO. To limit the parameter ranges, differences between input and exemplar OF values  $\varphi - \varphi_i$ , which are computed inside QDGRNN, are divided by  $p_{f1}$  sample quantile of the absolute differences when negative and  $p_{f2}$  quantile when nonnegative. The new parameters  $p_{f1}$ ,  $p_{f2}$  of the neural network replace  $s_\varphi$ . The OF decrement  $\varepsilon_f$  is another input parameter. It's relative too because it's subtracted after the division. Pointwise distances  $\|\mathbf{x}_f - \mathbf{X}_i\|$  are divided by their own  $p_x$  sample quantile, which replaces  $s_x$ . Thus the mapping performed by QDGRNN is as follows:

$$(10) \quad \mathbf{x}^* = \frac{\sum_{i=1}^{N_e} C_i^{(\varphi x)} \mathbf{X}_i}{\sum_{i=1}^{N_e} C_i^{(\varphi x)}},$$

$$(11) \quad C_i^{(\varphi x)} = C_i^{(\varphi)} \cdot C_i^{(x)},$$

$$(12)$$

$$C_i^{(\varphi)} = \exp \left( \ln 0.5 \left( \frac{\varphi - \varphi_i}{Q(\langle |\varphi - \varphi_j| : j = \overline{1, N_e} \rangle, p_f(\varphi, \varphi_i))} - \varepsilon_f \right)^2 \right),$$

$$(13) \quad p_f(\varphi, \varphi_i) = \begin{cases} p_{f1} & \text{if } \varphi < \varphi_i, \\ p_{f2} & \text{if } \varphi \geq \varphi_i, \end{cases}$$

$$(14) \quad C_i^{(x)} = \exp \left( \ln 0.5 \left( \frac{\|\mathbf{x}_f - \mathbf{X}_i\|}{Q(\langle \|\mathbf{x}_f - \mathbf{X}_j\| : j = \overline{1, N_e} \rangle, p_x)} \right)^2 \right),$$

$$(15)$$

where  $\mathbf{x}^*$  is the output vector;  $\varphi$  is the input OF value;  $\mathbf{x}_f$  is the input focus value;  $C_i^{(\varphi)}$ ,  $C_i^{(x)}$  are the similarity coefficients between the inputs and the corresponding exemplars  $\varphi_i$ ,  $\mathbf{X}_i$ ;  $C_i^{(\varphi x)}$  is the merged similarity

coefficient of the  $i$ th pair of exemplars;  $N_e$  is the number of exemplar pairs;  $Q(A, p)$  is the  $p$  quantile estimation for the sample  $A$ .

Parameter control is based on evolutionary principles. The parameters are set by *rules*, which crossover, mutation and selection operators are applied to. The population consists of  $N_r$  rules. A rule contains application conditions (not used in this work), measures of fitness and a tuple of parameters (genotype):  $\varepsilon_f, p_{f1}, p_{f2}, p_x, \alpha_b, P_r$ .

Fitness of each rule is measured by *charm* and *merit*. Charm is a short-term measure and regulates selection of rules for reproduction and application. Merit is a long-term measure. It's based on an exponential moving average of charm and determines survival of an individual.

This two-fold system preserves potentially useful individuals from extinction due to short-term situation changes. While currently efficient rules are actively reproducing and being applied, rules that became inefficient recently are "sleeping" waiting for an opportunity to become efficient again.

Iterations of the algorithm are divided into *big*, where application, evaluation and evolution of all rules take place, and *small*, where only some rules selected by charm are applied. The first and every  $I_{big}$ th iteration are big.

The population is initialized randomly or loaded from a saved rule base.

At each big iteration all rules are applied to the predefined number  $N_{top}$  of BPs. At the  $i$ th big iteration the BPs with numbers from  $1 + (i - 1) \cdot N_{top}$  to  $i \cdot N_{top}$  are selected. The count continues from the start when it goes over the end. For every unselected BP a randomly chosen rule is applied such that the selection probability is proportional to  $k_{sel}^r$ , where  $k_{sel} \in (0, 1)$  is a parameter and  $r$  is the zero-based rank of the rule in descending order of charm. Hence, all rules are periodically evaluated for various BPs, but rules with greater charm are applied more frequently.

*Application* of a rule at the BP  $\mathbf{b}$  consists of the following steps:

(1) First, an *attached set*  $P_b$  of probe points is formed:

$$(16) \quad P_b = \{ \mathbf{b} + i \cdot \mathbf{p} : i = \overline{-N_s, N_s}, \mathbf{p} \in P_r \},$$

where  $P_r$  is the *pattern* of the rule;  $\mathbf{p}$  is a *pattern vector*;  $N_s$  is an algorithm parameter.

(2) QDGRNN is trained to map OF values to coordinates on the attached set, then it maps BP's OF value:

$$(17) \quad \mathbf{b}^* = \text{QDGRNN}(\Phi(\mathbf{b}), \mathbf{b}, \varepsilon_f, p_{f1}, p_{f2}, p_x).$$



BP's position is the focus of QDGRNN and  $\varepsilon_f, p_{f1}, p_{f2}, p_x$  are set by the rule. Let us call the point  $\check{\mathbf{b}} = \mathbf{b} + \alpha_b (\mathbf{b}^* - \mathbf{b})$  the *rule candidate*, where the parameter  $\alpha_b$  is set by the rule. The difference  $\rho = \Phi(\mathbf{b}) - \Phi(\check{\mathbf{b}})$  is the *candidate progress*.

Points from the attached set are possible solutions too.

After application of rules the private goal  $\mathbf{g}_i^{[k]}$  for each BP is determined. At each iteration it's chosen by the minimum OF value from the list:  $\mathbf{g}_i^{[k-1]}$ , the current rule candidates for the BP,  $\mathbf{b}^{[k]}$ . Finally, the BP moves towards the goal according to the PSO-like rule:

$$(18) \quad \mathbf{v}^{[k]} = \omega_i \mathbf{v}^{[k-1]} + \omega_l \mathbf{R}_1 \otimes (\mathbf{g}_i^{[k]} - \mathbf{b}^{[k]}) + \omega_g \mathbf{R}_2 \otimes (\mathbf{g}_g^{[k]} - \mathbf{b}^{[k]}),$$

$$(19) \quad \mathbf{b}^{[k+1]} = \mathbf{b}^{[k]} + \mathbf{v}^{[k]},$$

where  $k$  is the iteration number;  $\omega_i, \omega_l, \omega_g$  are parameters of the algorithm;  $\mathbf{g}_i^{[k]}$  is the private goal of the BP;  $\mathbf{g}_g^{[k]}$  is the group goal;  $\mathbf{R}_1, \mathbf{R}_2$  are vectors of random numbers uniformly distributed between 0 and 1;  $\otimes$  denotes componentwise vector product.

Rules are evaluated by comparison of their candidate progress for the selected BPs at big iterations. A single first place (maximum progress) means higher charm than any number of second places, etc. Moreover, a rule that produced only negative progresses has lower charm than a rule that produced at least one positive progress. To achieve this, the vector of scores  $(\theta_1, \dots, \theta_{2N_r})$  is associated with each rule at each big iteration:

$$(20) \quad \theta_i = \sum_{j=1}^{N_{top}} \delta_{s_j i}, \quad s_i = \begin{cases} r_i & \text{if } \rho_i \geq 0, \\ r_i + N_r & \text{if } \rho_i < 0, \end{cases}$$

where  $N_{top}$  is the number of selected BPs (an algorithm parameter);  $\delta_{ij}$  is Kronecker delta;  $\rho_i$  is the progress of the rule candidate for the  $i$ th selected BP;  $r_i$  is the zero-based rank of  $\rho_i$  in descending order for the  $i$ th selected BP.

Then rules are sorted in descending lexicographical order of their score vectors. Let  $r$  be the zero-based rank of a rule in this list, then the rule's charm

$$(21) \quad charm^{[k+1]} = k_{cd}^r,$$

where  $k_{cd} \in (0, 1)$  is an algorithm parameter.

Merit is calculated as follows:

$$(22) \quad merit^{[k+1]} = merit^{[k]} \cdot k_{md}^a + charm^{[k]} + ppe^{[k]},$$

where  $k_{md} \in (0, 1)$  is a parameter;  $a$  is the age of the rule in iterations (starts from zero and grows by one at the end of each iteration);  $ppe^{[k]}$  is the ratio of the average candidate progress of the rule to the maximum average progress over all rules.

New rules are created by random generation, mutation and crossover. The probability of a rule being selected as a parent is proportional to its charm.

The procedure of random rule generation is as follows. Each scalar rule parameter  $\beta$  is set to a random number that is uniformly distributed on  $\left[l_{rand}^{(\beta)}, u_{rand}^{(\beta)}\right]$ . The number of pattern vectors is chosen at random from 1 to  $D$ . To produce a pattern vector, uniformly distributed random numbers  $p_{01}, \dots, p_{0D} \in [-1, 1]$  are generated. Then the components of the pattern vector are  $p_i = k_{mxp} \min_{j=1, D} \left( \frac{U_j - L_j}{|p_{0j}|} \right) p_{0i}$ , where  $k_{mxp}$  is a coefficient equal to  $k_{mxp1}$  for the proportion  $k_{mxpf}$  of the total number of random rules and  $k_{mxp2}$  for others. Therefore, the pattern vector has at least one component equal in magnitude to  $k_{mxp} (U_j - L_j)$ , and others are less than or equal in magnitude. By using two values of  $k_{mxp}$  we allow for both small and big pattern vectors to be generated.

Mutation produces a new rule by random perturbation of a parent rule's parameters. Akin to evolutionary programming, in the discussed algorithm mutation creates a new individual, so it's asexual reproduction. Mutation sets a scalar parameter of an offspring to a random value having a truncated normal distribution

$$(23) \quad \beta_o \sim \mathcal{N}_{l_{mut}, +\infty}^{(\beta)} \left( \beta_p, \sigma_{mut}^{(\beta)} \right),$$

where  $\beta_o, \beta_p$  are the values of the parameter of the offspring and of the parent respectively;  $\mathcal{N}_{l, u}(\mu, \sigma)$  is a truncated normal distribution with the mean  $\mu$  and the standard deviation  $\sigma$  on  $[l, u]$ ;  $l_{mut}^{(\beta)}, \sigma_{mut}^{(\beta)}$  are algorithm parameters defined independently for each rule parameter  $\beta$ .

Pattern vectors are distributed normally:

$$(24) \quad p_{o, ij} \sim \mathcal{N} \left( p_{p, ij}, \sigma_{mut}^{(p)} (U_j - L_j) \right),$$

where  $p_{o, ij}, p_{p, ij}$  are the  $j$ th components of the  $i$ th pattern vectors of the offspring and of the parent respectively;  $\mathcal{N}(\mu, \sigma)$  is a normal distribution with the mean  $\mu$  and the standard deviation  $\sigma$ ;  $\sigma_{mut}^{(p)}$  is an algorithm parameter.

Rules produced by mutation are tested for viability. An offspring is eliminated if any of its parameters  $p_{f1}, p_{f2}, p_x$  is less than or equal to zero.

Crossover of scalar parameters is defined by the formula  $\beta_o = u\beta_{p1} + (1 - u)\beta_{p2}$ , where  $\beta_o$ ,  $\beta_{p1}$ ,  $\beta_{p2}$  are the parameter values of the offspring, first and second parents respectively,  $u \in [0, 1]$  is a random uniformly distributed coefficient determined independently for each parameter. Crossover of two pattern vectors is performed as componentwise scalar crossover described above. The vectors are selected at random, one from the first parent and one from the second. The offspring pattern size is a random variable following the binomial distribution  $B(n, 0.5)$ , where  $n$  is the sum of the pattern sizes of the parents.

A new population of  $N_r$  rules is formed at each big iteration. First,  $N_{nr} = \lfloor (1 - k_{elt})N_r \rfloor$  new rules are created, among them  $\lfloor k_{rand}N_{nr} \rfloor$  random rules,  $\lfloor k_{cros}N_{nr} \rfloor$  crossover rules and  $\lfloor k_{mut}N_{nr} \rfloor$  mutation rules. Pattern vectors with norms less than  $\varepsilon_{pat}$  are removed from the new rules. Next, old rules are sorted in descending order of merit. First  $\lfloor k_{elt}N_r \rfloor$  rules from this list (the elite) are put into the new population. All new rules are checked for correctness. A rule is correct, if  $p_{f1}$ ,  $p_{f2}$ ,  $p_x$  are positive and the pattern is not empty. Incorrect rules are eliminated.

Local search and group goal trajectory extrapolation supplement the main algorithm.

Local search is performed at each group goal every  $I_{loc}$  iterations. In this work we used a modified BFGS [18, p. 136] algorithm. The algorithm stops after 100 iterations or when the norm of the gradient falls below  $10^{-12}$ . The last inverse Hessian approximation  $H_k$  computed by BFGS is saved and restored when local search is run next time. This memorized value becomes obsolete if the group goal shifts a lot between runs of local search. A modification of the algorithm done in this work allows for soft resetting of  $H_k$  in the following way: if the solution is not improved at the current iteration, then the next inverse Hessian approximation  $H_{k+1}$  is  $k_h H_k + (1 - k_h)E$ , where  $k_h \in [0, 1)$  is an algorithm parameter (equaled 0.75 in the experiments here),  $E$  is the identity matrix. The golden section method of line search is used that stops after 10 iterations or when Wolfe conditions with the coefficients  $c_1 = 10^{-4}$ ,  $c_2 = 0.9$  are satisfied [18, p. 33]. The line search restarts with the interval shortened 10 times if the OF value at the current point is greater than at the starting point.

The trajectory of every group goal is extrapolated each iteration. Let  $\mathbf{g}_g^{*[k]}$  be the group goal at the  $k$ th iteration,  $G_t^{[k]} = \{\mathbf{g}_g^{[k_1]}, \dots, \mathbf{g}_g^{[k_n]}\}$  be the set of its last  $n \leq N_{ext}$  best positions such that  $k > k_1 > \dots$  and  $\Phi(\mathbf{g}_g^{[k_1]}) < \dots < \Phi(\mathbf{g}_g^{[k_n]})$ , where  $N_{ext}$  is a parameter. Let  $P_{ext} = \left\{ \mathbf{g}_g^{*[k]} + \left( \mathbf{g}_g^{*[k]} - \mathbf{g}_g^{[k_i]} \right) : i = \overline{1, n} \right\}$ . Now the new group goal  $\mathbf{g}_g^{[k]}$  equals

$\arg \min_{\mathbf{x} \in \{\mathbf{g}_g^*[k]\} \cup P_{ext}} \Phi(\mathbf{x})$ .

The search continues until the stopping conditions are satisfied. They're checked each  $I_{stop}$  iterations. The algorithm stops when any of the listed conditions are satisfied: 1) the average decrease in the minimum OF value over the last  $I_{stop}$  iterations is less than  $\varepsilon_{stop}$ , 2) the average shift of the best solution point over the last  $I_{stop}$  iterations is less than  $\delta_{stop}$ ; 3) the number of iterations is greater than  $I_{max}$ . When the search stops, the resulting population is saved as a rule base that can be loaded again next time.

### 3. Experiments

To evaluate efficiency of the method, test minimization problems of 100 variables were solved ( $D = 100$ ). They were based on well-known test objective functions [19]. In all of the problems, the global minimum is at  $(0, 0, \dots)$  with the OF value of 0.

The search spaces are from  $-100$  to  $100$  for each coordinate by default. For some problems, the spaces are ones frequently used in the literature. As for Ackley's function, its local oscillations completely hide the global tendency far from the global minimum, so efficiency of the discussed method in this case essentially depends on the space size. Hence, two spaces were used: from  $-100$  to  $100$  and from  $-50$  to  $50$ .

The test problems are as follows:

(1) Rastrigin's function on  $[-100; 100]^D$ :

$$f_1(\mathbf{x}) = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10].$$

(2) Shifted Rosenbrock's function on  $[-100; 100]^D$ :

$$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} \left[ 100 \left( (x_{i+1} + 1) - (x_i + 1)^2 \right)^2 + x_i^2 \right].$$

(3) Ackley's function on  $[-100; 100]^D$ :

$$f_3(\mathbf{x}) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + e.$$

(4) Ackley's function on  $[-50; 50]^D$ .

(5) Griewangk's function on  $[-600; 600]^D$ :

$$f_4(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1.$$

(6) Weierstrass' function on  $[-0.5; 0.5]^D$ :

$$f_5(\mathbf{x}) = \sum_{i=1}^D \left[ \sum_{k=0}^{k_{max}} a^k \cos(2\pi b^k (x_i + 0.5)) \right] - D \sum_{k=0}^{k_{max}} a^k \cos(\pi b^k),$$

where  $a = 0.5$ ,  $b = 3$ ,  $k_{max} = 20$ .

(7) Katsuura's function on  $[-100; 100]^D$ :

$$f_6(\mathbf{x}) = 10D^{-2} \prod_{i=1}^D \left( 1 + i \sum_{j=1}^{32} |2^j x_i - \text{round}(2^j x_i)| 2^{-j} \right)^{10D^{-1.2}} - 10D^{-2},$$

where  $\text{round}(x)$  is  $x$  rounded to the closest integral value.

(8) Shifted Levy's function on  $[-100; 100]^D$ :

$$f_7(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] \\ + (w_D - 1)^2 [1 + \sin^2(2\pi w_D)],$$

where  $w_i = 1 + 0.25x_i$ .

(9) Sphere function on  $[-100; 100]^D$ :

$$f_8(\mathbf{x}) = \sum_{i=1}^D x_i^2.$$

(10) Disk function on  $[-100; 100]^D$ :

$$f_9(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^D x_i^2.$$

(11) Bent cigar function on  $[-100; 100]^D$ :

$$f_{10}(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2.$$

(12) Sum of different power function on  $[-100; 100]^D$ :

$$f_{11}(\mathbf{x}) = \sum_{i=1}^D |x_i|^{i+1}.$$

(13) High conditioned elliptic function on  $[-100; 100]^D$ :

$$f_{12}(\mathbf{x}) = \sum_{i=1}^D 10^{6(i-1)/(D-1)} x_i^2.$$

Classic test functions frequently have “convenient” properties: the global minimum is at zero or another point with equal coordinates or it can be found by searching along each coordinate independently:

$$(25) \quad \mathbf{x}_{min} = \left( \arg \min_{L_1 \leq x \leq U_1} \Phi(x, x_2, \dots), \arg \min_{L_2 \leq x \leq U_2} \Phi(x_1, x, \dots), \dots \right).$$

Therefore, random shifts and random orthogonal transformations are applied to all basic OFs, that is  $f_i^*(\mathbf{x}) = f_i(M(\mathbf{x} - \mathbf{x}_0))$ , where  $\mathbf{x}_0 = \mathbf{U}[\mathbf{L} + 0.4(\mathbf{U} - \mathbf{L}), \mathbf{L} + 0.6(\mathbf{U} - \mathbf{L})]$ . The matrix  $M$  is produced by Gram–Schmidt orthonormalization of a random matrix with elements uniformly distributed between 0 and 1. A similar approach was used in [19].

Experiments of the following types were conducted:

(1) Control experiments with random variation of parameters. The algorithm was run for each problem 100 times with random initialization. The rule base,  $M$ ,  $\mathbf{x}_0$  were initialized randomly before each run.

The described above NNAICM-PSO algorithm with disabled evolution was used. Charm and merit were set to random uniformly distributed numbers between 0 and 1 at each rule evaluation. The population was restricted to random rules without mutation or crossover.

(2) Experiments to evaluate general efficiency of NNAICM-PSO. The algorithm was run for each problem 100 times with random initialization. The rule base,  $M$ ,  $\mathbf{x}_0$  were initialized randomly before each run.

(3) Experiments to explore long-term adaptation capability of NNAICM-PSO. For each problem, 50 series by 10 runs of the algorithm were performed. The rule base was initialized randomly before the first run in series, saved after each run and reloaded for the next run (the *reused rule base* mode). Initialization of  $M$ ,  $\mathbf{x}_0$  was random and took place before the first and then each  $R_t$  runs in series. Experiments were conducted for  $R_t = 0$  ( $M$ ,  $\mathbf{x}_0$  don't change inside series) and  $R_t = 1$ .

For each run, the minimum OF value ( $\Phi_{min}$ ), the number of OF evaluations ( $N_{eval}$ ), the number of iterations ( $N_{iter}$ ) and the number of OF evaluations per iteration ( $N_{epi} = N_{eval}/N_{iter}$ ) were registered.

We use random variation as a control because it's known [20] that sometimes random variation of optimization algorithm parameters can improve results over static parameters by itself. We implemented random variation through modification of NNAICM-PSO to limit the differences between configurations to how parameters are set.

The parameters in the experiments were set as follows:

- the parameters of population size and composition:  $N_r = 100$ ,  $k_{elt} = 0.25$ ,  $k_{cros} = 0.5$ ,  $k_{mut} = 0.25$ ,  $k_{rand} = 0.25$ ;
- the parameters of mutation:  $l_{mut}^{(\varepsilon_f)} = l_{mut}^{(p_{f1})} = l_{mut}^{(p_{f2})} = l_{mut}^{(p_x)} = 10^{-2}$ ,  $\sigma_{mut}^{(\varepsilon_f)} = \sigma_{mut}^{(p_{f1})} = \sigma_{mut}^{(p_{f2})} = \sigma_{mut}^{(p_x)} = 0.05$ ,  $l_{mut}^{(\alpha_b)} = 10^{-2}$ ,  $\sigma_{mut}^{(\alpha_b)} = 0.05$ ,  $\sigma_{mut}^{(p)} = 5 \times 10^{-3}$ ;
- the parameters of rule evaluation:  $k_{cd} = 0.95$ ,  $k_{md} = 0.9999$ ;
- the parameters of rule application:  $I_{big} = 10$ ,  $k_{sel} = 0.95$ ,  $N_{top} = 10$ ,  $N_s = 1$ ;
- the parameters of random rule generation:  $k_{mxp1} = 0.2$ ,  $k_{mxp2} = 10^{-6}$ ,  $k_{mxpf} = 0.5$ ,  $\varepsilon_{pat} = 10^{-6}$ ,  $l_{rand}^{(\varepsilon_f)} = 10^{-2}$ ,  $u_{rand}^{(\varepsilon_f)} = 10$ ,  $l_{rand}^{(p_{f1})} = l_{rand}^{(p_{f2})} = l_{rand}^{(p_x)} = 10^{-2}$ ,  $u_{rand}^{(p_{f1})} = u_{rand}^{(p_{f2})} = u_{rand}^{(p_x)} = 1$ ,  $l_{rand}^{(\alpha_b)} = 10^{-2}$ ,  $u_{rand}^{(\alpha_b)} = 10$ ;
- the parameters of local search:  $I_{loc} = 25$ ,  $k_h = 0.75$ ;
- the parameters of BPs:  $N_b = 100$ ,  $k_{v1} = 1$ ,  $S_{bg} = 10$ ,  $v_{min} = 10^{-5}$ ,  $\varepsilon_b = 5 \times 10^{-6}$ ,  $I_{rest} = 10$ ,  $\omega_i = 0.9$ ,  $\omega_l = 0.5$ ,  $\omega_g = 0.5$ ,  $N_{ext} = 10$ ;
- the parameters of stopping conditions:  $I_{stop} = 100$ ,  $\varepsilon_{stop} = 10^{-7}$ ,  $\delta_{stop} = 0$ ,  $I_{max} = \infty$ .

#### 4. Results

The statistics of the experiments with random variation (RV) and evolutionary control (EC) of parameters without reloading of the rule base (the *discarded rule base* mode) are shown in Tables 1 and 2 respectively.

The results of the RV and EC experiments with discarded rule base are compared in Figures 3–5 and in Table 3. In Figure 3 the results are shown as box plots: the box represents the range between the first ( $Q_1$ ) and third ( $Q_3$ ) quartiles, the orange line denotes the median, whiskers extend to values not more than  $1.5(Q_3 - Q_1)$  below  $Q_1$  or above  $Q_3$ , more distant values (outliers) are denoted by circles, the green triangle shows the mean. In Figures 4, 5 error bars denote standard error of the mean.

The two-tailed Mann–Whitney test was used to test significance of

TABLE 1. Results of the RV experiments

Problem	Parameter	Minimum	Median	Maximum	Average	Standard dev.
1	$\Phi_{min}^*$	196	326.8	448.7	329.6	58.45
	$N_{eval}$	$9.817 \times 10^6$	$1.238 \times 10^7$	$2.468 \times 10^7$	$1.224 \times 10^7$	$2.162 \times 10^6$
	$N_{iter}$	400	500	1000	490	87.75
2	$\Phi_{min}^*$	$1.691 \times 10^{-12}$	$3.744 \times 10^{-11}$	3.987	1.316	1.875
	$N_{eval}$	$7.356 \times 10^6$	$7.663 \times 10^6$	$1.257 \times 10^7$	$8.545 \times 10^6$	$1.262 \times 10^6$
	$N_{iter}$	300	300	500	340	50.99
3	$\Phi_{min}^*$	20	20	20	20	0.000 200 3
	$N_{eval}$	$4.451 \times 10^6$	$4.681 \times 10^6$	$1.206 \times 10^7$	$5.309 \times 10^6$	$1.41 \times 10^6$
	$N_{iter}$	200	200	500	228	58.45
4	$\Phi_{min}^*$	$1.315 \times 10^{-8}$	19.81	19.91	19.2	3.378
	$N_{eval}$	$4.42 \times 10^6$	$4.866 \times 10^6$	$1.771 \times 10^7$	$6.826 \times 10^6$	$2.828 \times 10^6$
	$N_{iter}$	200	200	700	285	113.5
5	$\Phi_{min}^*$	$1.221 \times 10^{-14}$	$3.847 \times 10^{-14}$	$2.244 \times 10^{-13}$	$5.116 \times 10^{-14}$	$3.622 \times 10^{-14}$
	$N_{eval}$	$4.86 \times 10^6$	$5.033 \times 10^6$	$5.277 \times 10^6$	$5.033 \times 10^6$	$7.935 \times 10^4$
	$N_{iter}$	200	200	200	200	0
6	$\Phi_{min}^*$	71.3	93.86	106.3	93.57	6.629
	$N_{eval}$	$1.421 \times 10^7$	$1.9 \times 10^7$	$5.104 \times 10^7$	$2.164 \times 10^7$	$7.776 \times 10^6$
	$N_{iter}$	600	800	2100	893	316.6
7	$\Phi_{min}^*$	0.017 05	0.0621	0.3598	0.079 02	0.061 23
	$N_{eval}$	$6.731 \times 10^6$	$1.134 \times 10^7$	$1.409 \times 10^7$	$1.059 \times 10^7$	$1.497 \times 10^6$
	$N_{iter}$	300	500	600	459	61.8
8	$\Phi_{min}^*$	4343	9905	$1.225 \times 10^4$	9622	1535
	$N_{eval}$	$4.692 \times 10^6$	$7.331 \times 10^6$	$9.822 \times 10^7$	$1.137 \times 10^7$	$1.193 \times 10^7$
	$N_{iter}$	200	300	3900	463	474.1
9	$\Phi_{min}^*$	$5.492 \times 10^{-16}$	$1.843 \times 10^{-15}$	$2.566 \times 10^{-15}$	$1.764 \times 10^{-15}$	$4.518 \times 10^{-16}$
	$N_{eval}$	$4.961 \times 10^6$	$5.255 \times 10^6$	$5.493 \times 10^6$	$5.252 \times 10^6$	$8.844 \times 10^4$
	$N_{iter}$	200	200	200	200	0
10	$\Phi_{min}^*$	$3.787 \times 10^{-7}$	$5.37 \times 10^{-7}$	$8.014 \times 10^{-7}$	$5.362 \times 10^{-7}$	$8.902 \times 10^{-8}$
	$N_{eval}$	$4.655 \times 10^6$	$4.913 \times 10^6$	$5.112 \times 10^6$	$4.909 \times 10^6$	$9.112 \times 10^4$
	$N_{iter}$	200	200	200	200	0
11	$\Phi_{min}^*$	$2.172 \times 10^{-9}$	$1.879 \times 10^{-7}$	$4.297 \times 10^{-6}$	$5.649 \times 10^{-7}$	$8.42 \times 10^{-7}$
	$N_{eval}$	$4.948 \times 10^6$	$5.17 \times 10^6$	$7.846 \times 10^6$	$5.609 \times 10^6$	$9.719 \times 10^5$
	$N_{iter}$	200	200	300	218	38.42
12	$\Phi_{min}^*$	$1.381 \times 10^{-10}$	$1.144 \times 10^{-7}$	0.000 75	$1.284 \times 10^{-5}$	$7.662 \times 10^{-5}$
	$N_{eval}$	$3.262 \times 10^7$	$5.563 \times 10^7$	$1.011 \times 10^8$	$5.803 \times 10^7$	$1.656 \times 10^7$
	$N_{iter}$	1100	1900	3500	1976	580.2
13	$\Phi_{min}^*$	$3.87 \times 10^{-7}$	$1.066 \times 10^{-6}$	$4.416 \times 10^{-6}$	$1.185 \times 10^{-6}$	$5.696 \times 10^{-7}$
	$N_{eval}$	$4.89 \times 10^6$	$5.096 \times 10^6$	$7.596 \times 10^6$	$5.137 \times 10^6$	$3.564 \times 10^5$
	$N_{iter}$	200	200	300	202	14

differences in  $\Phi_{min}^*$ ,  $N_{eval}$ ,  $N_{epi}^1$ . For each of the parameters separately,  $p$ -values were adjusted for multiple comparisons by the Holm–Bonferroni method<sup>2</sup>.

<sup>1</sup>The function `scipy.stats.mannwhitneyu` from SciPy 0.19.1 [21] was used.

<sup>2</sup>The function `p.adjust` with the parameter `method = "holm"` from R 3.4.4 [22] was used.



TABLE 2. Results of the EC experiments with discarded rule base

Problem	Parameter	Minimum	Median	Maximum	Average	Standard dev.
1	$\Phi_{min}^*$	$5.862 \times 10^{-14}$	$3.428 \times 10^{-13}$	54.72	15.68	17.44
	$N_{eval}$	$8.175 \times 10^6$	$9.86 \times 10^6$	$1.677 \times 10^7$	$1.072 \times 10^7$	$2.073 \times 10^6$
	$N_{iter}$	300	300	600	358	73.73
2	$\Phi_{min}^*$	$1.606 \times 10^{-12}$	$2.991 \times 10^{-11}$	3.987	1.037	1.749
	$N_{eval}$	$8.071 \times 10^6$	$9.312 \times 10^6$	$1.522 \times 10^7$	$1.002 \times 10^7$	$1.523 \times 10^6$
	$N_{iter}$	300	300	500	339	50.78
3	$\Phi_{min}^*$	20	20	20	20	0.0003196
	$N_{eval}$	$4.15 \times 10^6$	$5.458 \times 10^6$	$1.441 \times 10^7$	$6.26 \times 10^6$	$1.739 \times 10^6$
	$N_{iter}$	200	200	500	239	63.08
4	$\Phi_{min}^*$	$8.689 \times 10^{-9}$	19.76	19.93	17.58	6.182
	$N_{eval}$	$4.211 \times 10^6$	$7.423 \times 10^6$	$1.876 \times 10^7$	$7.969 \times 10^6$	$3.271 \times 10^6$
	$N_{iter}$	200	300	800	296	118.3
5	$\Phi_{min}^*$	$1.044 \times 10^{-14}$	$3.825 \times 10^{-14}$	$4.118 \times 10^{-13}$	$5.061 \times 10^{-14}$	$4.683 \times 10^{-14}$
	$N_{eval}$	$5.194 \times 10^6$	$6.228 \times 10^6$	$6.819 \times 10^6$	$6.193 \times 10^6$	$3.46 \times 10^5$
	$N_{iter}$	200	200	200	200	0
6	$\Phi_{min}^*$	39.17	73.61	87.24	72.53	9.259
	$N_{eval}$	$1.751 \times 10^7$	$2.312 \times 10^7$	$1.056 \times 10^8$	$3.031 \times 10^7$	$1.767 \times 10^7$
	$N_{iter}$	600	750	2900	956	498.7
7	$\Phi_{min}^*$	0.009948	0.06676	0.2953	0.08215	0.05262
	$N_{eval}$	$7.304 \times 10^6$	$1.177 \times 10^7$	$2.132 \times 10^7$	$1.175 \times 10^7$	$2.315 \times 10^6$
	$N_{iter}$	300	500	800	468	81.09
8	$\Phi_{min}^*$	4.268	9257	$1.178 \times 10^4$	7998	3726
	$N_{eval}$	$4.664 \times 10^6$	$1.064 \times 10^7$	$8.407 \times 10^7$	$1.332 \times 10^7$	$1.2 \times 10^7$
	$N_{iter}$	200	400	2800	479	421.5
9	$\Phi_{min}^*$	$4.741 \times 10^{-16}$	$1.195 \times 10^{-15}$	$2.149 \times 10^{-15}$	$1.236 \times 10^{-15}$	$4.056 \times 10^{-16}$
	$N_{eval}$	$5.094 \times 10^6$	$6.505 \times 10^6$	$7.379 \times 10^6$	$6.483 \times 10^6$	$4.42 \times 10^5$
	$N_{iter}$	200	200	200	200	0
10	$\Phi_{min}^*$	$3.54 \times 10^{-7}$	$5.437 \times 10^{-7}$	$7.794 \times 10^{-7}$	$5.491 \times 10^{-7}$	$8.739 \times 10^{-8}$
	$N_{eval}$	$4.857 \times 10^6$	$5.465 \times 10^6$	$6.317 \times 10^6$	$5.487 \times 10^6$	$3.104 \times 10^5$
	$N_{iter}$	200	200	200	200	0
11	$\Phi_{min}^*$	$1.617 \times 10^{-9}$	$2.423 \times 10^{-7}$	$7.641 \times 10^{-6}$	$5.306 \times 10^{-7}$	$9.742 \times 10^{-7}$
	$N_{eval}$	$5.512 \times 10^6$	$6.435 \times 10^6$	$1.014 \times 10^7$	$6.69 \times 10^6$	$9.355 \times 10^5$
	$N_{iter}$	200	200	300	209	28.62
12	$\Phi_{min}^*$	$8.579 \times 10^{-11}$	$3.468 \times 10^{-7}$	0.004809	$8.098 \times 10^{-5}$	0.0005427
	$N_{eval}$	$3.623 \times 10^7$	$5.498 \times 10^7$	$1.267 \times 10^8$	$6.038 \times 10^7$	$1.812 \times 10^7$
	$N_{iter}$	1100	1600	3800	1778	543.1
13	$\Phi_{min}^*$	$3.427 \times 10^{-7}$	$1.03 \times 10^{-6}$	$2.895 \times 10^{-6}$	$1.085 \times 10^{-6}$	$4.042 \times 10^{-7}$
	$N_{eval}$	$5.155 \times 10^6$	$5.95 \times 10^6$	$8.479 \times 10^6$	$6.033 \times 10^6$	$4.624 \times 10^5$
	$N_{iter}$	200	200	300	202	14

The difference in  $\Phi_{min}^*$  between the RV and EC experiments is significant at the 0.05 level (see Table 3). Significant differences were present in problems 1, 4, 6, 9. The minimum OF values for these problems were higher in the RV mode than in the EC mode (Figure 3). It's worthy to note that in problem 8 the adjusted  $p$ -value of 0.14 was greater than the significance level, but 15 values in the EC mode were less than 50 (with the

TABLE 3. Comparison of  $\Phi_{min}^*$  between the RV and EC experiments

Problem	$P$ -value	Adjusted $p$ -value
1	<0.0001	<0.0001*
2	0.0893	0.6850
3	0.0856	0.6850
4	0.0004	0.0042*
5	0.7545	1
6	<0.0001	<0.0001*
7	0.2785	1
8	0.0160	0.1444
9	<0.0001	<0.0001*
10	0.2433	1
11	0.9912	1
12	0.2009	1
13	0.3242	1

Note: \*) statistically significant results ( $p < 0.05$ ).

minimum of 4.268) and the minimum result in the RV mode was 4343.

In the EC mode  $N_{eval}$  was significantly ( $p < 0.05$ ) higher (in problems 2–11, 13) or lower (in problem 1) (Figure 4). There was no significant change in problem 12. In all problems  $N_{epi}$  was significantly higher in the EC mode (Figure 5).

The most substantial difference between the RV and EC modes was observed in problem 1 with Rastrigin’s function. There was simultaneous reduction of error, number of OF evaluations and iterations, which was not seen in other cases.

Therefore, the experimental results are consistent with the hypothesis that in some problems  $\Phi_{min}^*$  is lower with evolutionary control of NNAICM-PSO parameters than with random variation.

The statistics of the last run in series in the EC experiments with reused rule base with random transformation at the start ( $R_t = 0$ ) or before each run ( $R_t = 1$ ) are presented in Tables 4 and 5.

The results of the EC experiments with reused rule base are compared for  $R_t = 0$  and  $R_t = 1$  in Figures 6–9 and in Table 6. In Figures 7–9 colored areas denote standard error of the mean.

The two-tailed paired Wilcoxon signed-rank test was used to test significance of differences in  $\Phi_{min}^*$ ,  $N_{eval}$ ,  $N_{epi}$  between the first and the last run in series<sup>3</sup>. For each of the three parameters separately  $p$ -values were adjusted for multiple comparisons by the Holm–Bonferroni method.

<sup>3</sup>The function `scipy.stats.wilcoxon` with the parameter `alternative = 'two-sided'` from SciPy 0.19.1 [21] was used.

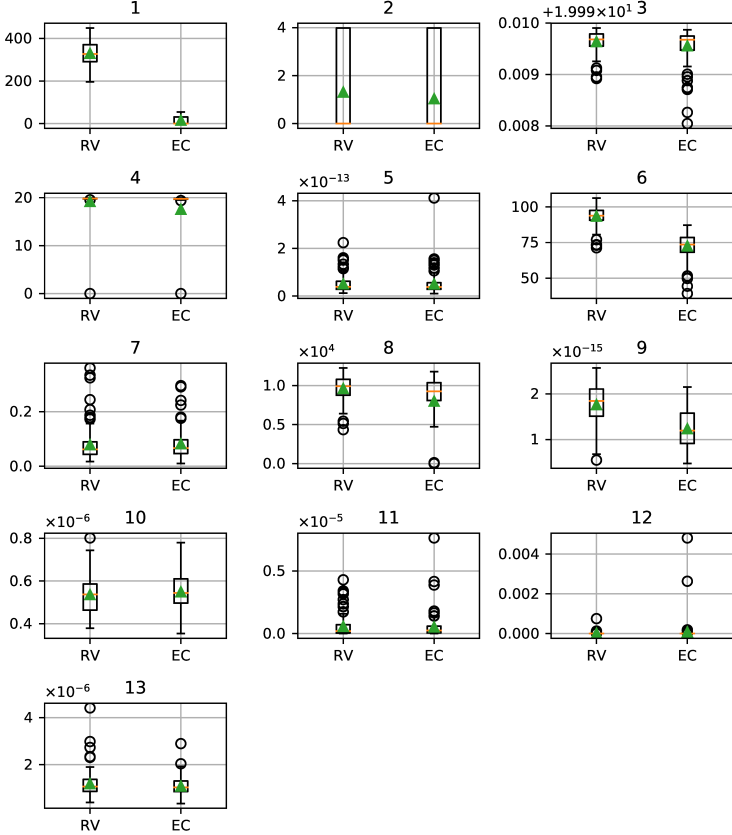


FIGURE 3.  $\Phi_{min}^*$  in the RV and EC experiments with discarded rule base

The difference in  $\Phi_{min}^*$  between the first and the last run in series is significant at the 0.05 level for  $R_t = 0$  and  $R_t = 1$  (see Table 6). In problems 1, 3, 4, 6, 8 the difference was present in both modes, while in problem 13 it was present in the  $R_t = 0$  mode only. In these problems  $\Phi_{min}^*$  obviously goes down in series (Figures 6, 7).

In most of the problems  $N_{eval}$  goes up in series (Figure 8). The differences between the first and the last run in series are significant ( $p < 0.05$ ) in problems 1, 2, 4, 5, 6, 9, 11, 13 with  $R_t = 0$  and 1, 2, 3, 4, 5, 7, 9, 11, 13 with  $R_t = 1$ .

In both modes, in all problems except No. 10,  $N_{epi}$  goes up in series (Figure 9). The differences in  $N_{epi}$  between the first and the last run in

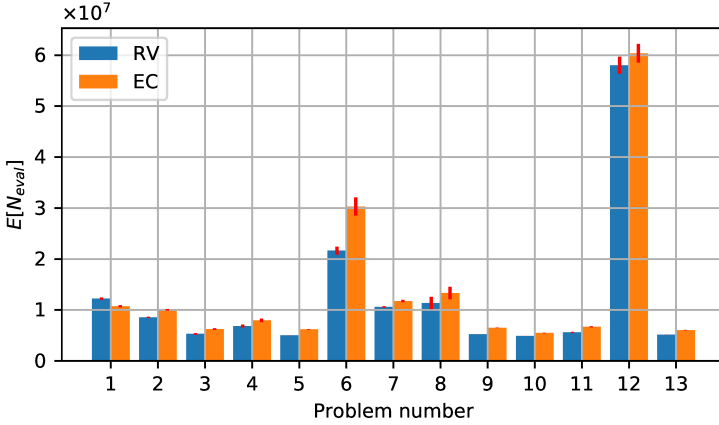


FIGURE 4. Average  $N_{eval}$  in the RV and EC experiments with discarded rule base

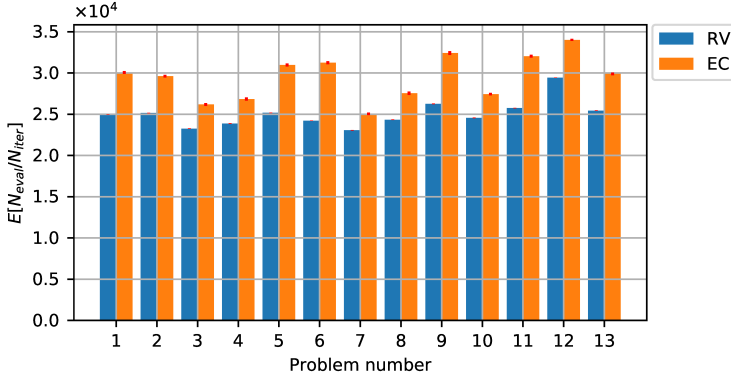


FIGURE 5. Average  $N_{epi}$  in the RV and EC experiments with discarded rule base

series are also significant ( $p < 0.05$ ) in all problems except No. 10. There are no significant differences in problem 10 in any mode.

No significant differences in  $\Phi_{min}^*$ ,  $N_{eval}$ ,  $N_{epi}$  between the  $R_t = 0$  and  $R_t = 1$  modes were discovered by the Mann–Whitney test ( $p > 0.05$ ).

Therefore, the experimental results are consistent with the hypothesis that in some problems evolutionary controlled NNAICM-PSO gives lower  $\Phi_{min}^*$  with reused rule base than with discarded rule base.

TABLE 4. Results of the last run in series in the EC experiments with reused rule base and  $R_t = 0$ 

Problem	Parameter	Minimum	Median	Maximum	Average	Standard dev.
1	$\Phi_{min}^*$	$6.395 \times 10^{-14}$	$9.415 \times 10^{-14}$	$1.634 \times 10^{-13}$	$1.006 \times 10^{-13}$	$2.451 \times 10^{-14}$
	$N_{eval}$	$1.075 \times 10^7$	$1.171 \times 10^7$	$1.234 \times 10^7$	$1.162 \times 10^7$	$4.48 \times 10^5$
	$N_{iter}$	300	300	300	300	0
2	$\Phi_{min}^*$	$2.191 \times 10^{-12}$	$3.458 \times 10^{-11}$	3.987	1.037	1.749
	$N_{eval}$	$9.954 \times 10^6$	$1.111 \times 10^7$	$1.675 \times 10^7$	$1.202 \times 10^7$	$1.882 \times 10^6$
	$N_{iter}$	300	300	500	340	56.57
3	$\Phi_{min}^*$	20	20	20	20	0.000 609 6
	$N_{eval}$	$4.734 \times 10^6$	$6.804 \times 10^6$	$1.28 \times 10^7$	$7.36 \times 10^6$	$2.116 \times 10^6$
	$N_{iter}$	200	200	400	258	75.07
4	$\Phi_{min}^*$	$8.239 \times 10^{-9}$	$1.779 \times 10^{-8}$	19.86	6.682	9.31
	$N_{eval}$	$5.553 \times 10^6$	$1.306 \times 10^7$	$2.688 \times 10^7$	$1.344 \times 10^7$	$4.848 \times 10^6$
	$N_{iter}$	200	400	800	414	154.9
5	$\Phi_{min}^*$	$1.232 \times 10^{-14}$	$3.531 \times 10^{-14}$	$2.001 \times 10^{-13}$	$4.908 \times 10^{-14}$	$4.164 \times 10^{-14}$
	$N_{eval}$	$6.779 \times 10^6$	$7.703 \times 10^6$	$8.103 \times 10^6$	$7.685 \times 10^6$	$3.045 \times 10^5$
	$N_{iter}$	200	200	200	200	0
6	$\Phi_{min}^*$	29.23	42.8	52.7	42.91	5.748
	$N_{eval}$	$2.017 \times 10^7$	$3.231 \times 10^7$	$7.287 \times 10^7$	$3.696 \times 10^7$	$1.242 \times 10^7$
	$N_{iter}$	500	800	1800	912	307
7	$\Phi_{min}^*$	0.018 59	0.063 55	0.2023	0.077 94	0.047 29
	$N_{eval}$	$7.554 \times 10^6$	$1.332 \times 10^7$	$2.502 \times 10^7$	$1.382 \times 10^7$	$3.14 \times 10^6$
	$N_{iter}$	400	500	800	472	84.95
8	$\Phi_{min}^*$	4573	7670	$1.141 \times 10^4$	7610	1625
	$N_{eval}$	$5.139 \times 10^6$	$1.515 \times 10^7$	$8.605 \times 10^7$	$1.873 \times 10^7$	$1.412 \times 10^7$
	$N_{iter}$	200	450	2200	570	392.6
9	$\Phi_{min}^*$	$5.241 \times 10^{-16}$	$1.192 \times 10^{-15}$	$2.145 \times 10^{-15}$	$1.198 \times 10^{-15}$	$3.635 \times 10^{-16}$
	$N_{eval}$	$7.097 \times 10^6$	$7.904 \times 10^6$	$8.587 \times 10^6$	$7.883 \times 10^6$	$3.433 \times 10^5$
	$N_{iter}$	200	200	200	200	0
10	$\Phi_{min}^*$	$3.2 \times 10^{-7}$	$4.975 \times 10^{-7}$	$8.597 \times 10^{-7}$	$5.253 \times 10^{-7}$	$1.154 \times 10^{-7}$
	$N_{eval}$	$4.731 \times 10^6$	$5.474 \times 10^6$	$6.203 \times 10^6$	$5.469 \times 10^6$	$3.423 \times 10^5$
	$N_{iter}$	200	200	200	200	0
11	$\Phi_{min}^*$	$2.003 \times 10^{-9}$	$2.199 \times 10^{-7}$	$1.65 \times 10^{-5}$	$8.341 \times 10^{-7}$	$2.444 \times 10^{-6}$
	$N_{eval}$	$7.014 \times 10^6$	$7.874 \times 10^6$	$1.222 \times 10^7$	$7.942 \times 10^6$	$6.694 \times 10^5$
	$N_{iter}$	200	200	300	202	14
12	$\Phi_{min}^*$	$7.689 \times 10^{-11}$	$2.102 \times 10^{-7}$	0.000 694 7	$1.686 \times 10^{-5}$	$9.707 \times 10^{-5}$
	$N_{eval}$	$3.901 \times 10^7$	$6.153 \times 10^7$	$1.137 \times 10^8$	$6.553 \times 10^7$	$1.751 \times 10^7$
	$N_{iter}$	1100	1750	3600	1800	523.5
13	$\Phi_{min}^*$	$4.134 \times 10^{-7}$	$9.624 \times 10^{-7}$	$1.968 \times 10^{-6}$	$9.968 \times 10^{-7}$	$3.303 \times 10^{-7}$
	$N_{eval}$	$5.578 \times 10^6$	$6.512 \times 10^6$	$7.56 \times 10^6$	$6.493 \times 10^6$	$4.409 \times 10^5$
	$N_{iter}$	200	200	200	200	0

## 5. Conclusion

Evolutionary control of parameters improves accuracy of the NNAICM-PSO method over random variation of parameters. The progressive decline of errors in objective function values in the reused rule base mode suggests adaptation of the method to the problem at hand.

TABLE 5. Results of the last run in series in the EC experiments with reused rule base and  $R_t = 1$

Problem	Parameter	Minimum	Median	Maximum	Average	Standard dev.
1	$\Phi_{min}^*$	$4.796 \times 10^{-14}$	$1.075 \times 10^{-13}$	$2.363 \times 10^{-13}$	$1.058 \times 10^{-13}$	$3.215 \times 10^{-14}$
	$N_{eval}$	$1.084 \times 10^7$	$1.162 \times 10^7$	$1.241 \times 10^7$	$1.158 \times 10^7$	$3.96 \times 10^5$
	$N_{iter}$	300	300	300	300	0
2	$\Phi_{min}^*$	$1.456 \times 10^{-12}$	$1.868 \times 10^{-11}$	3.987	0.3987	1.196
	$N_{eval}$	$9.828 \times 10^6$	$1.097 \times 10^7$	$1.804 \times 10^7$	$1.14 \times 10^7$	$1.534 \times 10^6$
	$N_{iter}$	300	300	500	316	41.76
3	$\Phi_{min}^*$	20	20	20	20	0.0004517
	$N_{eval}$	$4.5 \times 10^6$	$6.291 \times 10^6$	$1.371 \times 10^7$	$7.163 \times 10^6$	$2.146 \times 10^6$
	$N_{iter}$	200	200	400	252	64
4	$\Phi_{min}^*$	$7.62 \times 10^{-9}$	$2.076 \times 10^{-8}$	19.86	7.934	9.636
	$N_{eval}$	$5.842 \times 10^6$	$1.275 \times 10^7$	$2.632 \times 10^7$	$1.33 \times 10^7$	$5.439 \times 10^6$
	$N_{iter}$	200	400	700	416	151.5
5	$\Phi_{min}^*$	$8.216 \times 10^{-15}$	$3.014 \times 10^{-14}$	$1.402 \times 10^{-13}$	$4.073 \times 10^{-14}$	$2.919 \times 10^{-14}$
	$N_{eval}$	$7.109 \times 10^6$	$7.672 \times 10^6$	$8.296 \times 10^6$	$7.674 \times 10^6$	$2.754 \times 10^5$
	$N_{iter}$	200	200	200	200	0
6	$\Phi_{min}^*$	31.01	43.91	54.41	42.82	4.86
	$N_{eval}$	$2.425 \times 10^7$	$3.221 \times 10^7$	$6.958 \times 10^7$	$3.442 \times 10^7$	$9.846 \times 10^6$
	$N_{iter}$	600	800	1700	848	241.9
7	$\Phi_{min}^*$	0.01514	0.06195	0.2733	0.08082	0.05664
	$N_{eval}$	$7.564 \times 10^6$	$1.351 \times 10^7$	$2.24 \times 10^7$	$1.379 \times 10^7$	$3.266 \times 10^6$
	$N_{iter}$	300	500	800	468	88.18
8	$\Phi_{min}^*$	55.9	7049	$1.001 \times 10^4$	6852	2034
	$N_{eval}$	$5.144 \times 10^6$	$1.739 \times 10^7$	$6.373 \times 10^7$	$1.95 \times 10^7$	$1.118 \times 10^7$
	$N_{iter}$	200	600	1700	594	299.6
9	$\Phi_{min}^*$	$2.653 \times 10^{-16}$	$1.149 \times 10^{-15}$	$2.311 \times 10^{-15}$	$1.151 \times 10^{-15}$	$4.224 \times 10^{-16}$
	$N_{eval}$	$6.766 \times 10^6$	$7.846 \times 10^6$	$8.454 \times 10^6$	$7.831 \times 10^6$	$3.881 \times 10^5$
	$N_{iter}$	200	200	200	200	0
10	$\Phi_{min}^*$	$3.488 \times 10^{-7}$	$5.145 \times 10^{-7}$	$8.542 \times 10^{-7}$	$5.298 \times 10^{-7}$	$9.497 \times 10^{-8}$
	$N_{eval}$	$4.412 \times 10^6$	$5.495 \times 10^6$	$6.564 \times 10^6$	$5.498 \times 10^6$	$3.806 \times 10^5$
	$N_{iter}$	200	200	200	200	0
11	$\Phi_{min}^*$	$3.557 \times 10^{-9}$	$3.093 \times 10^{-7}$	$9.728 \times 10^{-6}$	$7.629 \times 10^{-7}$	$1.571 \times 10^{-6}$
	$N_{eval}$	$6.495 \times 10^6$	$7.887 \times 10^6$	$1.2 \times 10^7$	$7.972 \times 10^6$	$8.765 \times 10^5$
	$N_{iter}$	200	200	300	206	23.75
12	$\Phi_{min}^*$	$2.277 \times 10^{-10}$	$2.427 \times 10^{-7}$	0.005974	0.0001237	0.0008358
	$N_{eval}$	$3.3 \times 10^7$	$6.262 \times 10^7$	$1.433 \times 10^8$	$6.693 \times 10^7$	$2.334 \times 10^7$
	$N_{iter}$	900	1700	4200	1854	689.1
13	$\Phi_{min}^*$	$2.684 \times 10^{-7}$	$8.417 \times 10^{-7}$	$2.753 \times 10^{-6}$	$9.329 \times 10^{-7}$	$4.336 \times 10^{-7}$
	$N_{eval}$	$5.121 \times 10^6$	$6.492 \times 10^6$	$7.811 \times 10^6$	$6.53 \times 10^6$	$4.678 \times 10^5$
	$N_{iter}$	200	200	200	200	0

With evolutionary control and reusing of the rule base, there is no significant difference in results between random transformation of the problem at the start ( $R_t = 0$ ) or before each run of the algorithm in series ( $R_t = 1$ ). This indicates that the process is not sensitive to orthogonal transformations and limited shifts of the objective function.

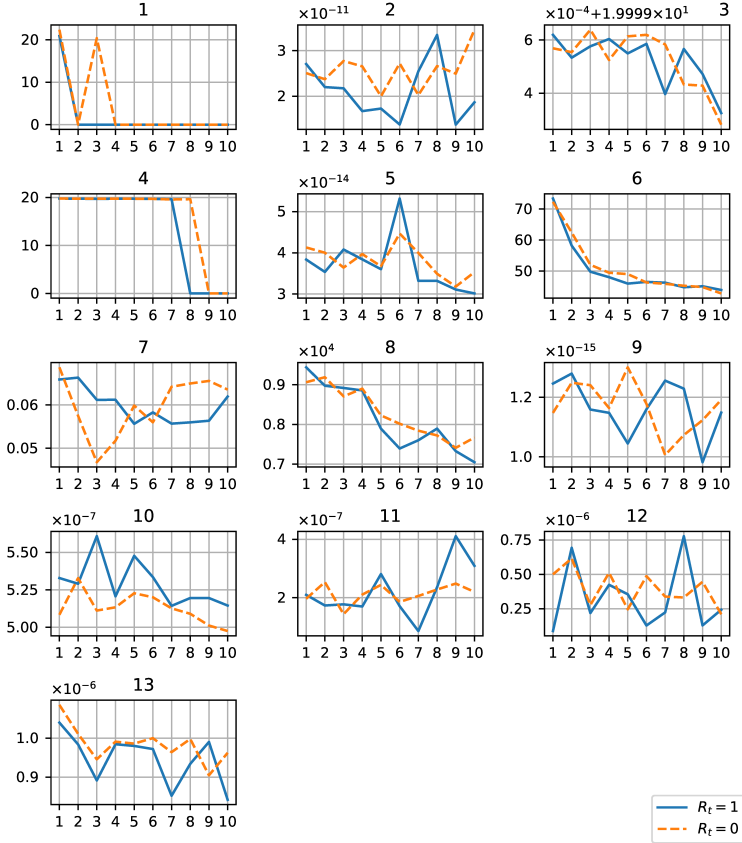


FIGURE 6. Median  $\Phi_{min}^*$  plotted against run number in series in the EC experiments with reused rule base

For random variation of parameters, the lowest average error was observed in problems 5, 9–13 and the lowest median error in problem 2. Only in problem 9 of these evolutionary control significantly reduced error. Moreover, there was no difference between the rule base being reused or discarded in these problems, except No. 13 with  $R_t = 0$ , during 10 runs of the algorithm. It's noteworthy that in all these problems the objective functions are polynomial or almost polynomial, including quadratic. For example, transformed Griewangk's function of 100 variables, used in problem 5, is quadratic in almost all of the search space except a small neighbourhood of the global minimum. High efficiency in such problems is

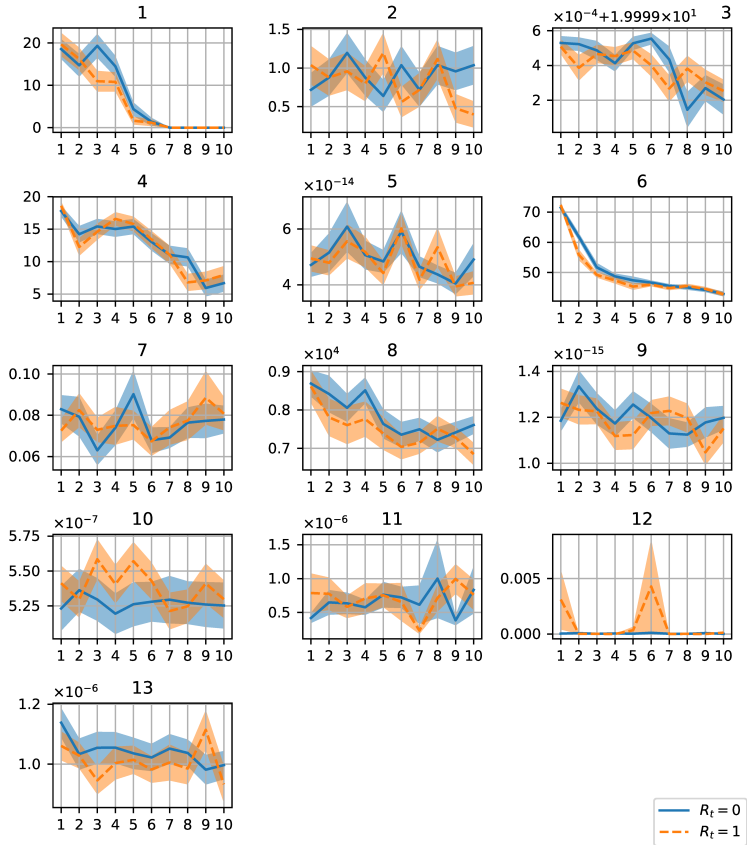


FIGURE 7. Average  $\Phi_{min}^*$  plotted against run number in series in the EC experiments with reused rule base

presumably due to supplementary quasi-Newton modified BFGS search.

The method worked well with transformed Rastrigin's function (problem 1), which is hard to minimize. Evolutionary control strongly reduced error in comparison to random variation of parameters in this case. In the reused rule base mode, average error quickly decreased from approximately 19 at the end of the first run to  $10^{-13}$  at the end of the 7th run in series (Figure 7). Results were also good for transformed Ackley's function (problem 4) in this mode, where in 10 runs the median error decreased from 20 to  $2 \times 10^{-8}$ .



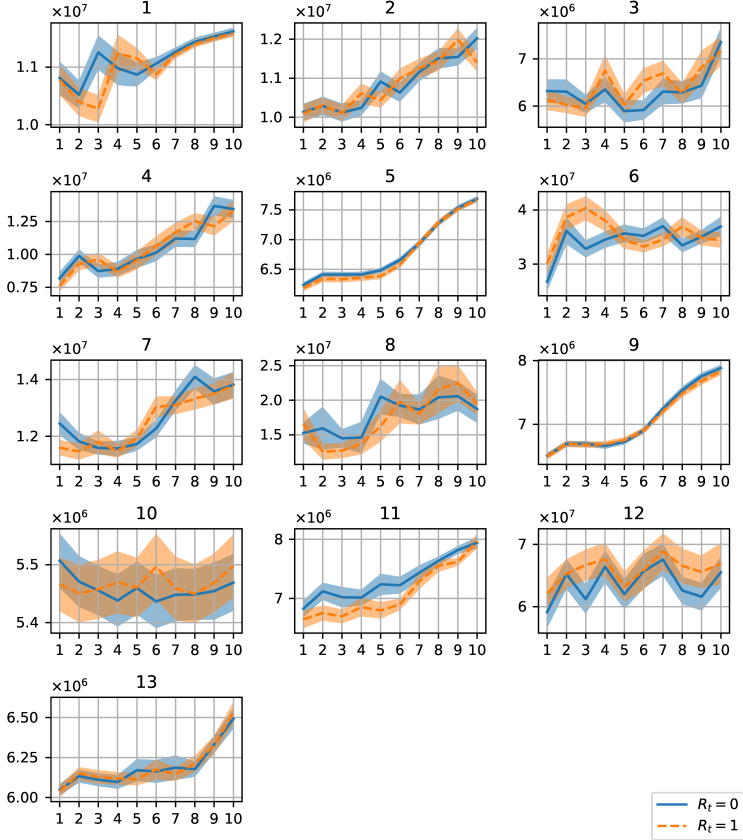


FIGURE 8. Average  $N_{eval}$  plotted against run number in series in the EC experiments with reused rule base

Statistical significance of the results was high. Usually the  $p$ -values were well below the threshold of 0.05.

Evolutionary control of NNAICM-PSO parameters has advantages over manual tuning. It allows to preserve, accumulate and reuse experience of solving a problem or problem class. Application of evolutionary control also allows us to exploit results from the field of evolutionary computation.

The evolutionary controller relieves the user of the burden of manual tuning of NNAICM-PSO parameters, although the ranges of random generation and mutation and other *metaparameters* of the controller itself

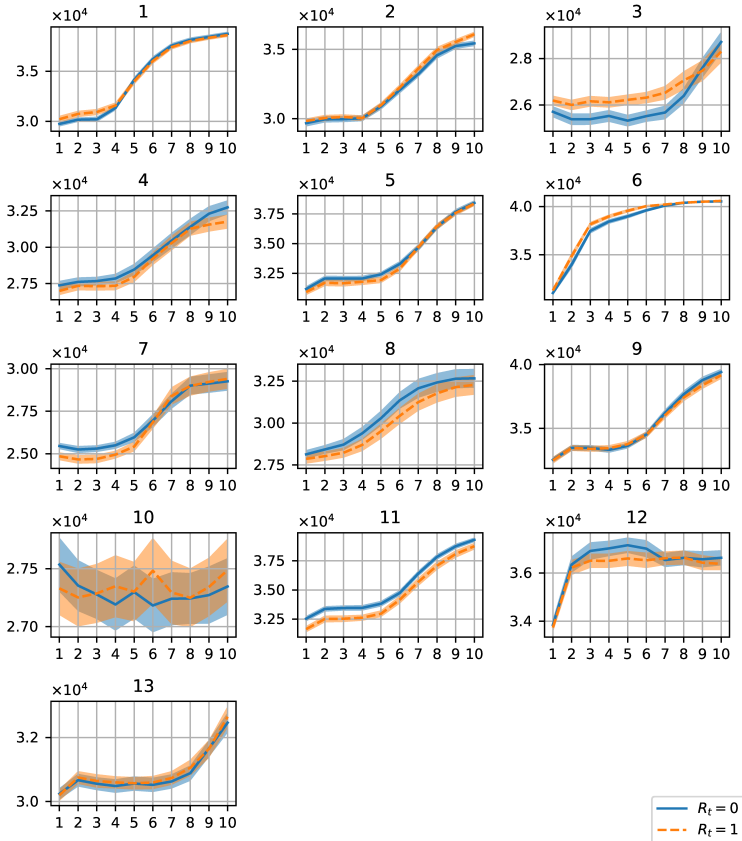


FIGURE 9. Average  $N_{epi}$  plotted against run number in series in the EC experiments with reused rule base

must still be tuned manually. Nevertheless, this task seems easier than direct manual tuning because, after initial metaparameter setting, the process will adapt to the problem. Suboptimal parameters can hinder evolution, but can't stop it. They can be corrected based on the direction of evolution deduced from the rule base.

The subjects of future study are (1) causes of efficiency or inefficiency of evolutionary controlled NNAICM-PSO in various situations, (2) how to reduce the number of objective function evaluations, (3) how to reduce the number of metaparameters.

TABLE 6. Comparison of  $\Phi_{min}^*$  between the first and the last run in series in the EC experiments with reused rule base




Problem	$R_t = 0$		$R_t = 1$	
	$P$ -value	Adjusted $p$ -value	$P$ -value	Adjusted $p$ -value
1	<0.0001	<0.0001*	<0.0001	<0.0001*
2	0.7030	1	0.1410	0.9333
3	0.0002	0.0022*	0.0012	0.0105*
4	<0.0001	<0.0001*	<0.0001	<0.0001*
5	0.5955	1	0.1333	0.9333
6	<0.0001	<0.0001*	<0.0001	<0.0001*
7	0.3466	1	0.6958	1
8	0.0022	0.0180*	0.0002	0.0022*
9	0.9961	1	0.2408	1
10	0.8734	1	0.4602	1
11	0.6191	1	0.5023	1
12	0.4840	1	0.9654	1
13	<0.0001	0.0002*	0.0341	0.2728

Note: \*) statistically significant results ( $p < 0.05$ ).

## References

- [1] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, H. De Garis. “An overview of evolutionary computation”, *European Conference on Machine Learning, Lecture Notes in Computer Science*, vol. **667**, Springer, 1993, pp. 442–459. doi ↑<sub>4</sub>
- [2] F. Neri, V. Tirronen. “Recent advances in differential evolution: a survey and experimental analysis”, *Artificial Intelligence Review*, **33**:1–2 (2010), pp. 61–106. doi ↑<sub>4</sub>
- [3] N. Siddique, H. Adeli. “Nature inspired computing: an overview and some future directions”, *Cognitive computation*, **7**:6 (2015), pp. 706–714. doi ↑<sub>4</sub>
- [4] D. H. Wolpert, W. G. Macready. “No free lunch theorems for optimization”, *IEEE transactions on evolutionary computation*, **1**:1 (1997), pp. 67–82. doi ↑<sub>4</sub>
- [5] G. Karafotias, M. Hoogendoorn, A. E. Eiben. “Parameter control in evolutionary algorithms: Trends and challenges”, *IEEE Transactions on Evolutionary Computation*, **19**:2 (2015), pp. 167–187. doi ↑<sub>4,5</sub>
- [6] A. Aleti, I. Moser. “A systematic literature review of adaptive parameter control methods for evolutionary algorithms”, *ACM Computing Surveys (CSUR)*, **49**:3 (2016), 56. doi ↑<sub>4,5</sub>
- [7] R. Poli, J. Kennedy, T. Blackwell. “Particle swarm optimization”, *Swarm intelligence*, **1**:1 (2007), pp. 33–57. doi ↑<sub>4</sub>
- [8] V. D. Koshur. “Reinforcement swarm intelligence in the global optimization method via neuro-fuzzy control of the search process”, *Optical Memory and Neural Networks*, **24**:2 (2015), pp. 102–108. doi ↑<sub>5</sub>

- [9] Sh. A. Akhmedova, V. V. Stanovov, E. S. Semenkin. “Cooperation of bio-inspired and evolutionary algorithms for neural network design”, *Journal of Siberian Federal University. Mathematics & Physics*, **11**:2 (2018), pp. 148–158. doi ↑<sub>5</sub>
- [10] E. Semenkin, M. Semenkina. “Self-configuring genetic algorithm with modified uniform crossover operator”, *Advances in Swarm Intelligence*, ICSI 2012, Lecture Notes in Computer Science, vol. **7331**, Springer, 2012, pp. 414–421. doi ↑<sub>5</sub>
- [11] G. Karafotias, A. E. Eiben, M. Hoogendoorn. “Generic parameter control with reinforcement learning”, *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ACM, 2014, pp. 1319–1326. doi ↑<sub>5</sub>
- [12] G. Karafotias, M. Hoogendoorn, B. Weel. “Comparing generic parameter controllers for EAs”, *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, IEEE, 2014, pp. 46–53. doi ↑<sub>5</sub>
- [13] A. Rost, I. Petrova, A. Buzdalova. “Adaptive Parameter Selection in Evolutionary Algorithms by Reinforcement Learning with Dynamic Discretization of Parameter Range”, *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, ACM, 2016, pp. 141–142. doi ↑<sub>5</sub>
- [14] V. Koshur, K. Pushkaryov. “Global optimization via neural network approximation of inverse coordinate mappings”, *Optical Memory and Neural Networks*, **20**:3 (2011), pp. 181–193. doi ↑<sub>5</sub>
- [15] V. D. Koshur, K. V. Pushkaryov. “Global’naya optimizatsiya na osnove neyrosetevoy approksimatsii inversnykh zavisimostey [Global optimization via neural network approximation of inverse mappings]”, XIII Vserossiyskaya nauchno-tehnicheskaya konferentsiya “Neyroinformatika-2011” [XIII All-Russian Scientific and Technical Conference “Neuroinformatics”], **1** (2010), pp. 89–98. ↑<sub>5</sub>
- [16] K. V. Pushkaryov, V. D. Koshur. “Gibridnyy evristicheskiy parallel’nyy metod global’noy optimizatsii [Hybrid heuristic parallel method of global optimization]”, *Vychislitel’nye metody i programmirovaniye [Computational Methods and Programming]*, **16** (2015), pp. 242–255. doi ↑<sub>5, 6, 7</sub>
- [17] V. D. Koshur, K. V. Pushkaryov. “Dual’nye obobshchenno-regressionnyye neyronnyye seti dlya resheniya zadach global’noy optimizatsii [Dual Generalized Regression Neural Networks for global optimization]”, XII Vserossiyskaya nauchno-tehnicheskaya konferentsiya “Neyroinformatika-2010” [XII All-Russian Scientific and Technical Conference “Neuroinformatics”], **2** (2010), pp. 219–227. ↑<sub>7</sub>
- [18] J. Nocedal, S. J. Wright. *Numerical Optimization*, Second Edition, Springer-Verlag, New York, 2006. doi ↑<sub>13</sub>
- [19] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, B. Y. Qu. “Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization”, 2016. URL ↑<sub>14, 16</sub>

- [20] G. Karafotias, M. Hoogendoorn, A. E. Eiben. “Why parameter control mechanisms should be benchmarked against random variation”, *IEEE Congress on Evolutionary Computation*, IEEE, 2013, pp. 349–355.  <sup>↑17</sup>
- [21] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.  <sup>↑18,20</sup>
- [22] R Core Team. *R: A language and environment for statistical computing*, 2018.  <sup>↑18</sup>


Received 27.04.2019  
 Revised 09.05.2019  
 Published 26.06.2019

Recommended by

*prof. Vyacheslav M. Khachumov*

*Sample citation of this publication:*

Kirill V. Pushkaryov. “Global optimization via neural network approximation of inverse coordinate mappings with evolutionary parameter control”. *Program Systems: Theory and Applications*, 2019, **10**:2(41), pp. 3–31.

 10.25209/2079-3316-2019-10-2-3-31

 [http://psta.psiras.ru/read/psta2019\\_2\\_3-31.pdf](http://psta.psiras.ru/read/psta2019_2_3-31.pdf)


The same article in Russian:  10.25209/2079-3316-2019-10-2-33-65

*About the author:*



### **Kirill Vladimirovich Pushkaryov**

Senior lecturer, Computer Science Department, Institute of Space and Information Technology, Siberian Federal University. Areas of interest include heuristic methods of global optimization and neural networks.

 ID 0000-0002-1138-886X

**e-mail:** [cyril.pushkaryov@yandex.ru](mailto:cyril.pushkaryov@yandex.ru)

Эта же статья по-русски:  10.25209/2079-3316-2019-10-2-33-65