JAISCR, 2020, Vol. 10, No. 3, pp. 151 – 171
10.2478/jaiscr-2020-0011

EVOLUTIONARY ALGORITHM WITH A CONFIGURABLE SEARCH MECHANISM

Krystian Łapa[1,*], Krzysztof Cpałka[1], Łukasz Laskowski[2], Andrzej Cader[3], Zhigang Zeng[4]

[1] *Czestochowa University of Technology, Department of Computational Intelligence, Częstochowa, Poland*

[2] *Polish Academy of Sciences, Institute of Nuclear Physics, Kraków, Poland*

[3] *Information Technology Institute, University of Social Sciences, Łódź, Poland and Clark University, Worcester, USA*

[4] *Huazhong University of Science and Technology, School of Automation, Wuhan, Hubei, China*

[*] *E-mail: krystian.lapa@pcz.pl*

Submitted: 5th September 2019; Accepted: 1st April 2020

Abstract

In this paper, we propose a new population-based evolutionary algorithm that automatically configures the used search mechanism during its operation, which consists in choosing for each individual of the population a single evolutionary operator from the pool. The pool of operators comes from various evolutionary algorithms. With this idea, a flexible balance between exploration and exploitation of the problem domain can be achieved. The approach proposed in this paper might offer an inspirational alternative in creating evolutionary algorithms and their modifications. Moreover, different strategies for mutating those parts of individuals that encode the used search operators are also taken into account. The effectiveness of the proposed algorithm has been tested using typical benchmarks used to test evolutionary algorithms.

Keywords: evolutionary algorithm, population-based algorithm, optimization, operator pool, operator selection, individual selection.

1 Introduction

Evolutionary algorithms that base on a population (PBAs) belong to meta-heuristic methods. These are methods of solving (usually optimization) problems that can be described by the concepts defined within them. They do not guarantee finding the optimal solution, but they are well suited for solving NP class problems. Normally, for such problems, the search time for finding an optimal solution is unacceptably long. The effectiveness of meta-heuristic algorithms depends on the effectiveness of the search mechanisms and their configuration (parameter selection).

PBAs are currently often used in real applications as the main and supporting methods alike. They are used in control [2, 25], modeling [5, 6, 26, 37], hardware applications [19, 30], identity verification [52]-[56], measuring systems [48], text recognition [44], etc. They also complement each other well with other Soft Computing methods [5, 9, 12, 28, 33, 34, 39, 42].

Table 1. The main features of the OP1 algorithm proposed in this paper.

Method	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**
Geem Z.W. et al. [17]	**yes**	no	no	no	no	no
Teodorovic D. et al. [45]	**yes**	no	no	**yes**	no	no
Chu S. et al. [7]	**yes**	no	**yes**	**yes**	no	no
Atashpaz-Gargari E. [4]	**yes**	no	**yes**	no	no	no
Yang X. [49]	**yes**	no	no	**yes**	no	no
Yang X., Deb S [50]	no	no	no	no	no	no
Rashedi E. [36]	**yes**	no	no	**yes**	no	no
Tan Y., Zhu Y. [43]	**yes**	no	**yes**	**yes**	no	no
Yang X. [51]	**yes**	**yes**	**yes**	no	no	no
Gandomi A.H., Alavi A.H. [15]	no	**yes**	**yes**	no	no	no
Wang B. et al. [46]	**yes**	**yes**	no	**yes**	no	no
Osaba E. et al. [35]	**yes**	**yes**	**yes**	**yes**	no	no
Mirjalili S. et al. [31]	**yes**	no	no	no	no	no
Łapa K. et al. [27]	**yes**	**yes**	**yes**	**yes**	no	no
Łapa K. et al. [29]	**yes**	**yes**	**yes**	**yes**	**yes**	no
proposed method	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**

f1 - Does the way the search mechanism works have an interpretation? **f2** - Does the way the search mechanism works fit into the problem under consideration? **f3** - Is the way the search mechanism operates assessed/evaluated and does it include the value of the evaluation function? **f4** - Can the search mechanism work differently for different individuals in the population? **f5** - Has the search engine working method been optimized with regard to the number of operators used? **f6** - Does the search mechanism provide flexible specialization of population individuals?

A multitude of applications is one of the factors affecting their intensive development and popularity. Currently, there are over one hundred different population algorithms, and each of them has many variations. Several conclusions follow from this fact. First of all, this field is becoming so extensive that it is difficult to find detailed comparative papers [20]. Existing reviews of population-based algorithms mainly concern specific methods and their variations [14]. Secondly, the development of new methods shows that universal and effective algorithms in various areas of application are still being sought [1]. Thirdly, the multitude of algorithm variants causes difficulties in choosing one for a specific application. It is also uncertain whether the solution found by one PBA could not be easily improved by using other PBAs. When selecting a PBA, it is also not possible to do it by the trial and error method because the number of PBAs is too high. Fourth, the emergence of new variations of algorithms may raise doubts about the desirability of such actions [40]. Therefore, an interesting research thread concerning PBAs may be creating general methods and using ensembles of different population-based algorithms [13]. This paper is specifically devoted to this topic. The algorithms proposed in it use known mechanisms of searching the solution search space and automatically adapt them to the problem under consideration. This is a new subject, but the solutions we proposed in our previous papers confirm that it is worth further investigation [27, 29].

1.1 Motivation

The primary mechanism of each PBA is the processing of individuals of a population. For this purpose dedicated operators are used. Some examples of such operators include the mutation operator from a Genetic Algorithm [38], and the wolf movement operator from the Grey Wolf Algorithm [31], etc.

If the operator's action on an individual is based on a small change in parameters, such operator is called an exploitation operator. However, if the operator's action is to look up a set of coded parameters in another part of the search space, the operator is then called the exploration operator. Some operators can balance smoothly between exploration and exploitation, or change the mode of operation depending on the distribution of individuals in the population [31]. Such operators can be included in both groups of operators. However, at the same time it is difficult to decide which operator/algorithm is suitable for solving a specific problem being considered.

The problem of proper balancing between exploration and exploitation is also considered in the literature [10, 32]. The effectiveness of PBAs also depends on it. This efficiency is understood as the appropriate quality of generated solutions, convergence and resistance to getting stuck in local optima. For this reason, many papers are published in which their authors create hybrids of different population-based algorithms. This increases the number of used operators and facilitates their temporary activation (depending e.g. on the evaluation of an individual). This is an interesting thread, but it allows one to create a virtually unlimited number of algorithm combinations. Its serious disadvantage is also the rigid connection of PBAs (and their operators) [3, 16, 21]. Furthermore, this method of hybridizing algorithms also causes problems typical for individual methods.

The authors' solutions are based on the use of a pool of evolutionary operators. Therefore, the method of modification of each individual depends on the operator associated with it. During the evolution of the population, individuals may change their operators. Of course, operators that achieve better optimization results can be automatically promoted, protected or propagated in the population. Thanks to this, the search mechanism and compromise between exploration and exploitation change dynamically and adapt to the problem and the temporary needs of the evolution process.

This topic was also dealt with in our previous works. Comments on the obtained results can be summarized as follows:

- **OPn** algorithm [27]. The algorithm used a pool of exploration and exploitation operators. They came from various PBAs. From this set, during its operation, the OPn selects a subset of n operators for each individual and they are used to generate new individuals of the population. The simulations presented in [27] showed that the average number of active operators used by the algorithm changed dynamically during the optimization process and ranged from 3 to 5. The algorithm presented in [27] was focused on designing the structure and parameters of the controller and its filters. For this reason, an important feature of the OPn method was that it was created by combining a genetic algorithm [18] and an algorithm based on the Particle Swarm Optimization (PSO) [24]. The purpose of the genetic algorithm was to select the structure of the controller, while the purpose of the PSO-based part was to optimize the parameters of the controller.
- **OP11** algorithm [29]. This algorithm used a pool of operators, similar to the OPn. However, unlike the OPn, it was based on the selection of a single operator for exploration and a single operator for exploitation for each individual. A certain disadvantage of the OP11 was the need to allocate each operator from the pool to one of the three groups: exploration, exploitation or universal. This defect, combined with the conclusions from the simulation, encouraged us to develop the method proposed in this paper.

1.2 Novelty elements

In this paper, we propose an approach in which a PBA can use a pool of operators from different PBAs. However, unlike the OPn [27] and OP11 [29] methods, it is based on the selection of a single evolutionary operator. This operator is flexibly selected from the pool of operators during the evolution process. The selection is made independently for each individual of the population and based on the value of its fitness function. This solution facilitates the specialization of individuals of the population, resulting from the use of appropriate operators. In the OPn and OP11 algorithms, the individuals of the population have universal characteristics.

Further on in this paper the basic version of the proposed algorithm will be called the OP1 for short (OP*erator-based algorithm with* 1 *search operator*). The features of this algorithm are shown in Table 1.

The original contribution of this paper can be summarized as follows:

- Describes the new OP1 algorithm that makes it possible to flexibly adjust the search space mechanism to the simulation problem. The use of one operator in the OP1 makes it easier to create a pool of operators that the OP1 uses - there is no need for each of the pool operators to specify its task: exploration or exploitation of the search space. This algorithm has not been previously presented in the literature.
- Describes the variation of the OP1 algorithm, which is characterized by a specific method of operator selection. It consists in the fact that in the initial phase of the evolution process, exploration operators have a better chance of being drawn. With each step of the algorithm's operation, the chance of random selection of exploitation operators increases linearly. In the basic version of the OP1, the probability of drawing any operator from the pool is constant at every

step of the evolution. This mechanism of operator selection has not been previously presented in the literature. The algorithm that uses this specific mechanism will be called the OP1L.

– Contains the simulation results presenting the dynamics of changes in the search mechanism that occur during the OP1 operation. In particular, it contains the results of the analysis of the average percentage use of the operators, which are included in Section 3.

– Describes the new operator selection method for PBAs that use the operator pool. It determines the strategy of operator exchange in the population and protects the operators of those individuals that have a favorable value of the fitness function. This approach has not been previously presented in the literature.

1.3 Structure of the paper

In Section 2 the proposed OP1 algorithm is described, Section 3 presents the simulation results and in Section 4 the conclusions are drawn.

2 Description of the OP1 algorithm

Comments on the OP1 algorithm proposed in this paper can be summarized as follows:

– In the OP1, each individual of population $\mathbf{X}_{ch}$ ($ch = 1, 2, \ldots, Nind$) has been extended with the use of additional parameter $\mathbf{X}^{\mathrm{op}}_{ch} = \left\{X^{\mathrm{op}}_{ch,1}\right\}$ (see Figure 1). It encodes the search operator. This solution makes the OP1 less computationally complex than the OP11 and OPn algorithms, which simultaneously used a larger number of operators.

– In the OP1, a set of used operators (derived from different PBAs) was placed in a common pool (see Figure 1) without distinguishing between exploration, exploitation and universal operators. They are indexed as follows $\mathrm{op}_1, \mathrm{op}_2, \ldots, \mathrm{op}_{Nop}$. This facilitates the implementation of the operator pool compared to the solution used in the OP11 [29].

– The OP1 algorithm is based on the way the PSO algorithm works [24] (see Algorithm 1). Thus, in each individual an additional set of parameters (besides $\mathbf{X}^{\mathrm{op}}_{ch}$ and problem parameters $\mathbf{X}^{\mathrm{par}}_{ch}$) representing the best-found-so-far solution encoded in $\mathbf{X}^{\mathrm{bst}}_{ch}$ and individual's velocity $\mathbf{X}^{\mathrm{vel}}_{ch}$ is added (interpreted as in the PSO [24]). In the next steps of the algorithm, the velocity of each $\mathbf{X}_{ch}$ can be gradually decreased depending on the operators used.

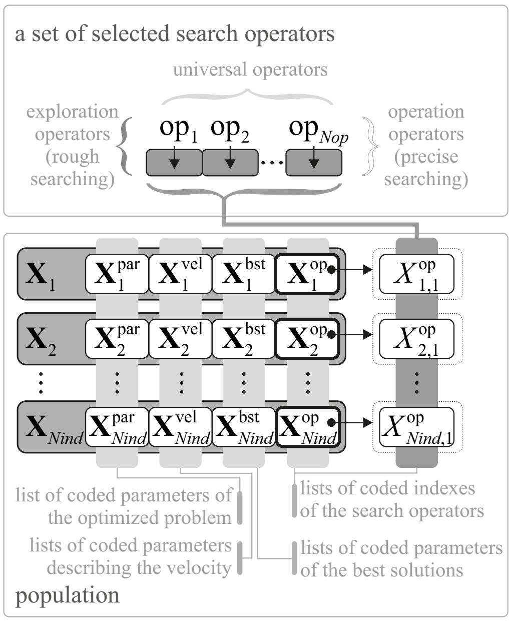
2.1 Individual's structure

In the OP1 a single individual's structure $\mathbf{X}_{ch}$ from population $\mathbf{P}$ is defined as follows

$$\begin{aligned} \mathbf{X}_{ch} &= \left\{\mathbf{X}^{\mathrm{par}}_{ch}, \mathbf{X}^{\mathrm{vel}}_{ch}, \mathbf{X}^{\mathrm{bst}}_{ch}, \mathbf{X}^{\mathrm{op}}_{ch}\right\} \\ &= \left\{X_{ch,1}, \ldots, X_{ch,Nc}\right\}, \end{aligned} \tag{1}$$

where Nc stands for the total number of the parameters in individual $\mathbf{X}_{ch}$, $\mathbf{X}^{\mathrm{par}}_{ch}$ encodes problem parameters, $\mathbf{X}^{\mathrm{vel}}_{ch}$ encodes velocity parameters $\mathbf{X}_{ch}$, $\mathbf{X}^{\mathrm{bst}}_{ch}$ encodes the best solution found by individual $\mathbf{X}_{ch}$, $\mathbf{X}^{\mathrm{op}}_{ch} = \left\{X^{\mathrm{op}}_{ch,1}\right\}$ encodes indexes of search operators (see Fig 1 and Table 2).

Further on in the paper, it was assumed that notation '$\mathbf{X}_{ch} : \mathbf{X}^{\mathrm{bst}}_{ch}$' refers to component $\mathbf{X}^{\mathrm{bst}}_{ch}$ encoded in individual $\mathbf{X}_{ch}$ ($ch = 1, 2, \ldots, Nind$).



a set of selected search operators
universal operators
exploration operators (rough searching)
op1 op2 opNop
operation operators (precise searching)
X1 X1par X1vel X1bst X1op X1,1op
X2 X2par X2vel X2bst X2op X2,1op
XNind XNindpar XNindvel XNindbst XNindop XNind,1op
list of coded parameters of the optimized problem
lists of coded parameters describing the velocity
lists of coded indexes of the search operators
lists of coded parameters of the best solutions
population

Figure 1. The structure of population individuals adopted in the OP1 algorithm.

Table 2. A set of the OP1 operators selected for the simulation purposes. Additional functions and notations used by these operators are shown in Table 3. Additional parameters of the operators are shown in Table 4.

i	**Base operator** (type)	$op_i(X^{\text{par}}_{ch,g})$
1	PSO-global [23] (global)	$c_1 \cdot \mathrm{U}(0,1) \cdot \left(X^{\text{glb}}_{g} - X^{\text{par}}_{ch,g}\right)$
2	GA-crossover [11] (global)	$\begin{cases} \mathrm{U}(0,1) \cdot \left(X^{\text{par}}_{ch1,g} - X^{\text{par}}_{ch,g}\right) \text{ for } \mathrm{Uind}(0,1) < p_c \\ 0 \text{ otherwise} \end{cases}$
3	DE-crossover [41] (global)	$\begin{cases} F \cdot \left(X^{\text{par}}_{ch1,g} - X^{\text{par}}_{ch2,g}\right) \text{ for } (\mathrm{Uind}(0,1) < CR) \text{ or } (ch = \mathrm{Rind}) \\ 0 \text{ otherwise} \end{cases}$
4	FFA-movement [49] (global)	$\begin{cases} \beta_0 \cdot \left(X^{\text{par}}_{ch1,g} - X^{\text{par}}_{ch,g}\right) \cdot e^{-\gamma \cdot (\mathbf{X}_{ch1}, \mathbf{X}_{ch})^2} \text{ for } \mathrm{ff}(\mathbf{X}_{ch1}) < \mathrm{ff}(\mathbf{X}_{ch}) \\ 0 \text{ otherwise} \end{cases}$
5	CS-lévy flights [50] (global)	$\alpha \cdot \mathrm{L}(s, \lambda)$
6	ABC-candidate [22] (global)	$\begin{cases} \mathrm{U}(-1,1) \cdot \left(X^{\text{par}}_{ch1,g} - X^{\text{par}}_{ch,g}\right) \text{ for } ch = \mathrm{RInd} \\ 0 \text{ otherwise} \end{cases}$
7	BAT-walk [51] (global+local)	$\mathrm{U}(-1,1) \cdot \sum_{i=1}^{Nind} \frac{At}{Nind}$
8	FWA-explosion [43] (global+local)	$\begin{cases} \frac{\mathrm{U}(-1,1) \cdot \hat{A} \cdot \left(\mathrm{ff}\left(\mathbf{X}_{ch}\right) - \mathrm{ff}_{min}\right)}{\sum_{i=1}^{Nind} (\mathrm{ff}(\mathbf{X}_i) - \mathrm{ff}_{min})} \text{ for } ch \in Rset \\ 0 \text{ otherwise} \end{cases}$
9	FWA-mutation [43] (global+local)	$\begin{cases} X^{\text{par}}_{ch,g} \cdot \mathrm{Ug}(1,1) - X^{\text{par}}_{ch,g} \text{ for } ch \in Rset \\ 0 \text{ otherwise} \end{cases}$
10	GWO-move [31] (global+local)	$\frac{1}{3} \cdot \left(\begin{array}{c} X^{\alpha}_{ch,g} + X^{\beta}_{ch,g} + X^{\gamma}_{ch,g} - A_1 \cdot \left\lvert C_1 \cdot X^{\alpha}_{ch,g} - X^{\text{par}}_{ch,g} \right\rvert + \\ -A_2 \cdot \left\lvert C_2 \cdot X^{\beta}_{ch,g} - X^{\text{par}}_{ch,g} \right\rvert - A_3 \cdot \left\lvert C_3 \cdot X^{\gamma}_{ch,g} - X^{\text{par}}_{ch,g} \right\rvert \end{array} \right)$
11	PSO-best [23] (local)	$c_2 \cdot \mathrm{U}(0,1) \cdot \left(X^{\text{bst}}_{ch,g} - X^{\text{par}}_{ch,g}\right)$
12	GA-mutation [11] (local)	$\begin{cases} \mathrm{U}(-1,1) \cdot m_r \text{ for } \mathrm{U}(0,1) < p_m \\ 0 \text{ otherwise} \end{cases}$
13	BAT-movement [51] (local)	$\mathrm{U}(f_{\min}, f_{\min} + f_{\max}) \cdot \left(X^{\text{glb}}_{g} - X^{\text{par}}_{ch,g}\right)$
14	FFA-walk [49] (local)	$\alpha \cdot \mathrm{U}(-0.5, 0.5)$
15	CS-random walk [50] (local)	$\alpha \cdot s \cdot \mathrm{H}(p_a - \mathrm{U}(0,1)) \cdot \left(X^{par}_{ch1,g} - X^{par}_{ch2,g}\right)$
16	BTO-history [8] (local)	$F^{\text{BTO}} \cdot \mathrm{U}(0,1) \cdot \left(X^{par}_{ch,g} - h_i\right)$

Table 3. Functions and notations used by the operators of the OP1 algorithm considered in the simulations. The set of these operators is shown in Table 2. The Table assumes that function rnd returns a random value from the range $\langle 0,1\rangle$ and D is the dimension of the simulation problem.

Symbol	**Definition or notation**	**Description**
$\mathrm{U}(a,b)$	$a+\mathrm{rnd}\cdot(b-a)$	real random number from the range $\langle a,b\rangle$
$\mathrm{Uind}(a,b)$	$a+\mathrm{rnd}\cdot(b-a)$	$\mathrm{U}(a,b)$ generated once during a single modification of an individual
$ch1, ch2$	$\langle 1, Nind\rangle$	indexes of individuals selected using the roulette wheel selection method [38]
Rind	$\mathrm{round}(\mathrm{rnd}\cdot D)$	random index of the parameter encoding the solution
$\mathrm{dist}(\mathbf{X}_a,\mathbf{X}_b)$	$\frac{1}{D}\cdot\sum_{h=1}^{D}\left\lvert X_{a,h}^{par}-X_{b,h}^{par}\right\rvert$	distance between individuals $\mathbf{X}_a$ and $\mathbf{X}_b$
$\mathrm{L}(s,\lambda)$	$\mathrm{L}(s,\lambda)=\frac{\lambda\cdot\Gamma(\lambda)\cdot\sin(0.5\cdot\pi\cdot\lambda)}{\pi\cdot s^{1+\lambda}}$	Lévy distribution (λ was set to 1)
$\Gamma(x)$	$\Gamma(x+1)=x\cdot\Gamma(x)$, $\Gamma(1)=1$	Euler's gamma
At	$\begin{cases} 1 \text{ for } i=1 \\ \alpha\cdot At \text{ for } i>1 \end{cases}$	bat loudness
$Rset$	the selection method described in [43]	a set of indexes of individuals being fireworks [43]
$\mathrm{ff}(\mathbf{X}_{ch})$	the definition depends on the problem	the value of the evaluation function of an individual
ff_{min}	$\min\{\mathrm{ff}(\mathbf{X}_1),...,\mathrm{ff}(\mathbf{X}_{Nind})\}$	the smallest value of the evaluation function of the individual from the entire population
$\mathrm{Ug}(1,1)$	$\frac{1}{\sqrt{2\cdot\pi}}\cdot e^{-\frac{1}{2}\cdot(\mathrm{U}(-2,4)-1)^2}$	Gaussian distribution with mean 1 and standard deviation 1 for FWA
$\mathbf{X}^{\alpha},\mathbf{X}^{\beta},\mathbf{X}^{\gamma}$	the best individuals of the population	values of parameters of the best individuals of the population (wolves from the herd)
$[C_1,C_2,C_3]$	$C_i=2\cdot\mathrm{U}(0,1)$	coefficient vector 1 for GWO
$[A_1,A_2,A_3]$	$A_i=2\cdot a\cdot\mathrm{U}(0,1)-a$	coefficient vector 2 for GWO
a	$-\frac{2}{Nsteps}\cdot step+2$	component linearly decreasing from 2 to 0 over iterations ($step=1,2,...,Nsteps$)
$\mathrm{H}(x)$	$\frac{d}{dx}\max\{x,0\}$	Heaviside function
h_i	$\mathrm{permuting}(h_i)$	historical population [8]

Table 4. Parameters of the OP1 algorithm operators used in the simulations. The set of the selected operators is shown in Table 2. The range of values was determined on the basis of the guidelines given in the literature.

Parameter	Range	Value	Description
w	$\langle 0.80, 1.00\rangle$	0.80	the constriction factor (PSO [23])
c_1	$\langle 1.50, 2.50\rangle$	2.00	social parameter (PSO [23])
F	$\langle 0.00, 1.00\rangle$	0.50	mutation factor (DE [41])
CR	$\langle 0.00, 1.00\rangle$	0.50	crossover constant (DE [41])
β_0	$\langle 0.90, 1.10\rangle$	1.00	attractiveness at step 0 (FFA [49])
γ	$\langle 0.50, 2.00\rangle$	1.25	attractiveness constant (FFA [49])
s	$\langle 0.00, 0.30\rangle$	0.15	step size (CS [50])
α	$\langle 0.10, 0.30\rangle$	0.20	scaling factor (CS [50])
λ	$\langle 0.00, 1.00\rangle$	0.50	switching parameter (CS [50])
α	$\langle 0.90, 1.00\rangle$	0.95	loudness constant (BAT [51])
$\hat{A}$	$\langle 0.10, 2.00\rangle$	1.05	amplitude constant (FWA [43])
c_2	$\langle 1.50, 2.50\rangle$	2.00	cognitive parameter (PSO [23])
p_m	$\langle 0.05, 0.50\rangle$	0.25	mutation probability (GA [11])
p_c	$\langle 0.50, 0.90\rangle$	0.70	crossover probability (GA [11])
m_r	$\langle 0.01, 0.20\rangle$	0.10	mutation range (GA [11])
f_{min}	$\langle 0.00, 0.50\rangle$	0.25	minimum wavelength (BAT [51])
f_{max}	$\langle 0.00, 1.00\rangle$	0.50	maximum wavelength (BAT [51])
α	$\langle 0.01, 0.20\rangle$	0.10	walk constant (FFA [49])
p_a	$\langle 0.00, 1.00\rangle$	0.50	switching parameter (CS [50])
F^{BTO}	$\langle 3.00, 4.00\rangle$	3.50	step size amplification (BTO [8])

2.2 Modification of individuals

Modification of parameters in the OP1 is different for components $\{\mathbf{X}_{ch}^{\mathrm{par}}, \mathbf{X}_{ch}^{\mathrm{vel}}\}$ storing real parameters and $\mathbf{X}_{ch}^{\mathrm{op}}$ storing integral parameter. The method of determining parameters $\mathbf{X}_{ch}^{\mathrm{bst}}$ was shown in Algorithm 1 (lines 13-15).

Modification of parameters $\mathbf{X}_{ch}^{\mathrm{par}}$ and $\mathbf{X}_{ch}^{\mathrm{vel}}$ of individual $\mathbf{X}_{ch}$ is as follows

$$\begin{cases} X_{ch,g}^{\mathrm{vel}} := w \cdot X_{ch,g}^{\mathrm{vel}} + \mathrm{op}_{X_{ch,1}^{\mathrm{op}}}\left(X_{ch,g}^{\mathrm{par}}\right) \\ X_{ch,g}^{\mathrm{par}} := X_{ch,g}^{\mathrm{par}} + X_{ch,g}^{\mathrm{vel}} \end{cases}, \tag{2}$$

where w stands for the inertia weight (usually from range $w \in \langle 0.8, 1.0\rangle$ [23]), g ($g = 1, 2, \ldots, D$) stands for the index of the component in individual $\mathbf{X}_{ch}$, D stands for the search space dimension (related to the simulation problem), $\mathrm{op}_i(\cdot)$ stands for search operator i (see Table 2).

Modification of parameter $\mathbf{X}_{ch}^{\mathrm{op}}$ consists in the crossover and mutation of its value. The crossover of parameter $\mathbf{X}_{ch}^{\mathrm{op}}$ is performed when condition $\mathrm{Uind}(0,1) < p_c$ for individual ch is met. Because the crossover does not create new individuals in the population, but only applies to the modification of the operator selection parameter, the likelihood of such a change has been set to a small value. Function $\mathrm{Uind}(min, max)$ returns the real random number from the range $\langle min, max\rangle$, $p_c \in (0,1)$ means the probability of the crossover of an individual $\mathbf{X}_{ch}$. If individual ch meets the given condition (ie. $\mathrm{Uind}(0,1) < p_c$), then the second individual for crossover is selected in the algorithm by the selection method. The considered selection methods are described in Section 2.4. Crossover involves assigning parameter $X_{ch,1}^{\mathrm{op}}$ from a selected parent individual. Approaches to $\mathbf{X}_{ch}^{\mathrm{op}}$ mutation are described in Section 2.5.

2.3 Evolution process

The OP1 algorithm works according to the scheme shown in Algorithm 1. The algorithm begins its operation with a random initialization of population $\mathbf{P}$ (line 1). This procedure takes into account the ranges of the parameters encoded in individuals $\mathbf{X}_{ch}$ ($ch = 1, 2, \ldots, Nind$) of population $\mathbf{P}$. Next, population $\mathbf{P}$ is evaluated using a problem-dependent fitness function (line 2). From initialized and evaluated population $\mathbf{P}$ an individual $\mathbf{X}^{\mathrm{glb}}$

with the best value of fitness function $\mathrm{ff}(\cdot)$ is selected (line 3). Moreover, for each $\mathbf{X}_{ch}$ ($ch = 1,2,\ldots,Nind$) the copy of their parameters $\mathbf{X}^{\mathrm{bst}}_{ch}$ is stored, which will be updated in the next steps of the algorithm (lines 4-6). Then, the algorithm goes to the iterative part (lines 7-20), which starts with checking the stop condition. The stop condition may include, for example, taking a specified number of steps ($Nsteps$), executing an allowable number of fitness function calls $\mathrm{ff}(\cdot)$ or reaching by $\mathrm{ff}\left(\mathbf{X}^{\mathrm{glb}}\right)$ a specified threshold. In this part of the algorithm, a modification of population $\mathbf{P}$ is performed according to Section 2.2 (lines 9-10). After $\mathbf{P}$ has been modified, it is sometimes necessary to check and correct the values of individual parameters $\mathbf{P}$ taking into account the permissible ranges of changes (line 11). Then, takes place the evaluation of $\mathbf{P}$ (line 12), a component $\mathbf{X}^{\mathrm{bst}}_{ch}$ ($ch = 1,2,\ldots,Nind$) is updated (lines 13-15) and actualization of $\mathbf{X}^{\mathrm{glb}}$ (lines 16-18) is done. After that, the algorithm returns to checking the stop condition (line 7). If it is met, the algorithm presents the solution encoded in $\mathbf{X}^{\mathrm{glb}}$ (line 21) and terminates its operation.

Algorithm 1 Algorithm OP1 in the case of the problem of minimizing the value of fitness function $\mathrm{ff}(\cdot)$.

```
1: initialization of $Nind$ individuals of $\mathbf{P}$
2: evaluation of population $\mathbf{P}$ with the use of fitness
   function $\mathrm{ff}(\cdot)$
3: selection of best individual $\mathbf{X}^{\mathrm{glb}}$ from $\mathbf{P}$
4: for $ch := 1$ to $Nind$ do
5:     $\mathbf{X}_{ch} : \mathbf{X}^{\mathrm{bst}}_{ch} := \mathbf{X}_{ch}$        ▷ actualization of $\mathbf{X}^{\mathrm{bst}}_{ch}$
6: end for
7: while stop condition is not met do
8:     for $ch := 1$ to $Nind$ do
9:         modification of $\mathbf{X}^{\mathrm{op}}_{ch}$ of individual $\mathbf{X}_{ch}$
10:        modification of $\mathbf{X}^{\mathrm{par}}_{ch}$ and $\mathbf{X}^{\mathrm{vel}}_{ch}$ of $\mathbf{X}_{ch}$
11:        repair of $\mathbf{X}_{ch}$
12:        evaluation of $\mathbf{X}_{ch}$ (ff $\left(\mathbf{X}_{ch}\right)$ calculation)
13:        if $\mathrm{ff}\left(\mathbf{X}_{ch}\right) < \mathrm{ff}\left(\mathbf{X}_{ch} : \mathbf{X}^{\mathrm{bst}}_{ch}\right)$ then
14:            $\mathbf{X}_{ch} : \mathbf{X}^{\mathrm{bst}}_{ch} := \mathbf{X}_{ch}$
15:        end if
16:        if $\mathrm{ff}\left(\mathbf{X}_{ch}\right) < \mathrm{ff}\left(\mathbf{X}^{\mathrm{glb}}\right)$ then
17:            $\mathbf{X}^{\mathrm{glb}} := \mathbf{X}_{ch}$
18:        end if
19:    end for
20: end while
21: presentation of the solution coded in component
    $\mathbf{X}^{\mathrm{par}}$ of individual $\mathbf{X}^{\mathrm{glb}}$
```

2.4 Selection of individuals

In step 10 of Algorithm 1 components $\mathbf{X}^{\mathrm{par}}_{ch}$ and $\mathbf{X}^{\mathrm{vel}}_{ch}$ of individual $\mathbf{X}_{ch}$ are modified, which involves an appropriate use of the operator indicated by component $\mathbf{X}^{\mathrm{op}}_{ch}$. Some operators (given in Table 2 and others) are based on several individuals of the population. Sometimes they are specific individuals (e.g. in the GWO method), but usually they should be selected accordingly. In the simulations we used typical methods for selecting individuals, i.e. the roulette wheel (RW), the tournament strategy (TS), and the rank strategy (RS) [38].

2.5 Mutation of component $X^{\mathrm{op}}_{ch,1}$

In the simulation, three approaches to the mutation of $X^{\mathrm{op}}_{ch,1}$ are used. In the first one, we assumed that in each step of the algorithm ($step = 1,2,\ldots Nsteps$), the probability of drawing any operator from the operator pool is the same

$$X^{\mathrm{op}}_{ch,1} = \mathrm{UI}(1, Nop) \text{ for } \mathrm{U}(0,1) < p_m, \tag{3}$$

where function $\mathrm{UI}(min, max)$ returns an integral random value from the range $\langle min, max\rangle$, $\mathrm{U}(min, max)$ stands for a real random number from the range $\langle min, max\rangle$, $p_m \in (0,1)$ stands for the mutation probability for individual $\mathbf{X}_{ch}$. This (static) type of mutation was noted as 'ST'.

In the second approach, an assumption was made that probability of mutation of $X^{\mathrm{op}}_{ch,1}$ depends on its fitness function value. Thus, a modified version of equation (3) takes the following form

$$\begin{array}{l} X^{\mathrm{op}}_{ch,1} = \mathrm{UI}(1, Nop) \text{ for} \\ \mathrm{U}(0,1) < \mathrm{Fpm}\left(\begin{array}{c} \mathrm{ff}\left(\mathbf{X}_{ch}\right), \\ pmBest, pmWorst, \\ \min\limits_{ch1=1,2,\ldots,Nind}\left(\mathrm{ff}\left(\mathbf{X}_{ch1}\right)\right), \\ \max\limits_{ch1=1,2,\ldots,Nind}\left(\mathrm{ff}\left(\mathbf{X}_{ch1}\right)\right) \end{array}\right), \end{array} \tag{4}$$

where $\{pmBest, pmWorst\}$ are the values for parameter p_m for the best and worst individuals (they are algorithm parameters) and $\mathrm{Fpm}(\cdot)$ is a linear function, which further ensures normalization of the value of the population fitness function to the range $\langle 0,1\rangle$. It is defined as follows

$$\mathrm{Fpm}\begin{pmatrix} ff, pmBest, pmWorst, \\ ffMin, ffMax \end{pmatrix} = pmBest + \frac{\begin{pmatrix} (ff - ffMin)\cdot \\ \cdot(pmWorst - pmBest) \end{pmatrix}}{ffMax - ffMin}, \tag{5}$$

where ff stands for the current value of the fitness function of the processed individual. This mutation variation, which is based on the relationship (4) and (linear) (5) was noted as 'LI'.

In the third approach, the entire population of individuals is divided into three groups: (a) individuals with the best value of the fitness function (with a count of $Nbest$), (b) individuals with the worst value of the fitness function (with a count of $Nworst$) and (c) other individuals (equal to $Npop - (Nbest + Nworst)$). All 3 groups of individuals use formula (3), but with a different value p_m: the first group with $p_m = 0$, the second group with $p_m = 1$, and the third group with p_m specified by the algorithm parameter.

The purpose of this approach is to protect component $X^{\mathrm{op}}_{ch,1}$ of those individuals that encode the best solutions. At the same time, the other components of individuals $\mathbf{X}_{ch}$ in form (1) are modified accordingly to formula (2). This type of mutation was noted as 'PR'. The use of this mutation also requires normalization of the vector components of the fitness function: $\mathrm{ff}(\mathbf{X}_{ch})$ $(ch = 1, 2, \ldots Nind)$: $(\mathrm{ff}(\mathbf{X}_{ch}) - ffMin)/(ffMax - ffMin)$ to range $\langle 0, 1\rangle$.

2.6 OP1L modification

The OP1L modification is based on the OP1. However, in the OP1L an assumption that the probability of drawing an operator from the operator pool depends on its mode of operation (exploration or exploitation). In this approach: (a) at the beginning of the evolution process, the chance of selecting exploration operators increases and (b) at the end of the evolution process, the chance of selecting exploitation operators increases. The purpose of this is to try to organize how the operators are randomly selected.

Practical implementation of the OP1L mutation formula replaces formula (3) with the following formula

$$X^{\mathrm{op}}_{ch,1} = \begin{cases} \mathrm{UI}(1, Nr + Nu) \text{ for} \\ \begin{pmatrix} \mathrm{U}_1(0,1) < p_m \text{ and} \\ \mathrm{U}_2(0,1) < \mathrm{Fpm}\begin{pmatrix} step, pmBest, \\ pmWorst, \\ 1, Nsteps \end{pmatrix} \end{pmatrix} \\ \mathrm{UI}(Nr + 1, Nr + Nu + Np) \text{ for} \\ \begin{pmatrix} \mathrm{U}_1(0,1) < p_m \text{ and} \\ \mathrm{U}_2(0,1) \geq \mathrm{Fpm}\begin{pmatrix} step, pmBest, \\ pmWorst, \\ 1, Nsteps \end{pmatrix} \end{pmatrix} \end{cases}, \tag{6}$$

where $step$ is a current step of the evolution ($step = 1, 2, \ldots Nsteps$, where $Nsteps$ stands for the total number of steps), Nr stands for the number of exploration operators (global), Np stands for the number of exploitation operators (local), and Nu stands for the number of universal operators (that can work as exploration and exploitation-see Table 2). Function $\mathrm{Fpm}(\cdot)$ used in equation (6) promotes exploration in the early stages of the algorithm and exploitation in the final stages of the evolution. It is a linear function of the following form

$$\mathrm{Fpm}\begin{pmatrix} step, pm_{start}, pm_{stop}, \\ Nsteps_{start}, Nsteps_{stop} \end{pmatrix} = \frac{\begin{pmatrix} step \cdot \begin{pmatrix} pm_{start} + \\ -pm_{stop} \end{pmatrix} + \\ + \begin{pmatrix} pm_{stop} \cdot Nsteps_{start} + \\ -pm_{start} \cdot Nsteps_{stop} \end{pmatrix} \end{pmatrix}}{Nsteps_{start} - Nsteps_{stop}}. \tag{7}$$

As was mentioned in Section 1.2, the algorithm that uses dependencies (6) and (7) is called the OP1L.

3 Simulations

The main purpose of the simulation was to compare the OP1 algorithm with the methods: the OPn [27], the OP11 [29] and other popular PBAs whose operators were used in the OP1 operator pool (Table 2).

In the simulations popular test functions from CEC2013 [47] were used. There are 28 functions, and the purpose of each PBA was to search for such arguments for which functions took a minimum. In all functions, the number of variables (dimension) was set to $D = 50$.

The assumptions adopted in the simulations are described in Section 3.1, the conclusions from the simulation are summarized in Section 3.2, and a comparison of the results obtained using different PBAs is presented in Section 3.3.

3.1 Simulations assumptions

Assumptions adopted in the simulations can be summarized as follows:

– The stop criterion for the tested PBAs was to perform 1000 steps. Each simulation was repeated 200 times and the results obtained were averaged.

– In the OP1 a pool of 16 operators was used. They are presented in Table 2, wherein $Nr = 6$, $Nu = 4$, and $Np = 6$ (see Section 1.3). In the notation of these operators, functions and symbols were used and they are defined in Table 3. Moreover, for each of these operators, typical (suggested in the literature) values of parameters were adopted. They are listed in Table 4.

– Three methods of selecting individuals were tested: RW, TS, and RS (see Section 2.4). The basic method was RW.

– Three approaches to mutations were tested: ST, LI, PR (see Section 2.5). The following parameters were adopted in them: $p_c = 0.2$, $p_m = 0.1$, $pmBest = 0.0$, $pmWorst = 0.2$, and $Nbest = Nworst = 25\%$.

3.2 Simulations conclusions

The conclusions from the simulations can be summarized as follows:

– A summary of standard and average values of fitness function $\mathrm{ff}(\cdot)$ for simulation problems C-01 - C-28 is shown in Tables 5-7. The best results were obtained for the case of OPn-RW-ST (Table 7, columns 4 and 5). The OP11 algorithm gave good results for the RW and PR cases (Table 7, rows RW-PR). The OP1 and OP1L algorithms allowed to obtain satisfactory results, but slightly worse than the OPn and OP11 (Table 7, column 5). However, it is difficult to find an algorithm that would generate the best solutions in terms of the value of the fitness function (Tables 5 and 6). For example, an average rating for OPn-RW-ST is 7.6 (Table 7, column 6). An interesting observation also concerns the number of operators used in the OPn. There were on average no more than 2.5 operators used (Table 7, column 7).

– A summary of the value of the fitness functions grouped by selection strategies and mutation strategies of $X^{\mathrm{op}}_{ch,1}$ is shown in Table 8. The RW selection method gave the best results (Table 8, column 'Sel' and 'Avg'). As for the mutation method it was the ST method that gave the best results (Table 8, column 'Mut' and 'Avg'). The mutation of the operators with protective thresholds gave the best results for the OP11 algorithm (Table 8, row 'OP11-PR'). For the other algorithms, the results obtained using PR are acceptable.

– A comparison of the average operator utilization by the best PBAs variants (see Table 10), for example, function C-10, is shown in Figure 2. The figure shows that some operators were not used (e.g. the DE-cross). The analysis of the use of operators also shows that the use of some of them increased with algorithm steps (e.g. the GA-cross), while others decreased (e.g. the FFA-move). The dynamics of changes in the use of operators confirm the need to replace them during the evolution process.

– A comparison of the average operator usage for C-05 - C-10 test functions and the OPn-RW-ST algorithm is shown in Figure 3. The figure shows that the number of operators used varies in different ways for different functions. Therefore, it is difficult to indicate a subset of operators that will be suitable for each test function. This behavior of the operator selection mechanism also confirms the ability of the OPn algorithm to adapt to the problem under consideration and the situation in the current step of the evolution process.

– A comparison of the average operator usage for C-05 - C-10 test functions and the OP1-TS-ST algorithm is shown in Figure 4. The figure

Table 5. Normalized and averaged values of fitness function ff$(\cdot)$ for C-01 - C-14 simulation problems.

Alg.	Sel.	Mut.	C-01	C-02	C-03	C-04	C-05	C-06	C-07	C-08	C-09	C-10	C-11	C-12	C-13	C-14
		ST	0.00	0.08	0.14	0.08	0.00	0.15	0.40	0.20	0.00	0.09	0.28	0.48	0.43	0.03
	RW	LI	0.00	0.23	0.15	0.15	0.00	0.18	0.65	0.00	0.37	0.05	0.36	0.51	0.37	0.05
		PR	0.00	0.53	0.34	0.40	0.01	0.22	1.00	0.70	0.60	0.10	0.58	0.64	0.60	0.38
		ST	0.00	0.04	0.07	0.04	0.00	0.24	0.33	0.18	0.13	0.16	0.38	0.50	0.43	0.00
OPn	TS	LI	0.00	0.11	0.10	0.09	0.00	0.34	0.39	0.37	0.37	0.10	0.37	0.60	0.51	0.09
		PR	0.01	0.38	0.26	0.37	0.02	0.35	0.68	0.64	0.53	0.18	0.70	0.76	0.69	0.31
		ST	0.00	0.00	0.15	0.00	0.17	0.37	0.62	0.11	0.27	0.23	0.56	0.75	0.64	0.05
	RS	LI	0.00	0.04	0.19	0.05	0.15	0.53	0.69	0.25	0.55	0.23	0.63	0.75	0.61	0.17
		PR	0.03	0.35	0.41	0.27	1.00	0.56	0.95	0.57	0.78	0.38	0.98	0.93	0.86	0.39
		ST	0.00	0.35	0.30	0.73	0.00	0.07	0.54	0.86	0.56	0.00	0.00	0.00	0.00	0.76
	RW	LI	0.00	0.32	0.15	0.48	0.00	0.14	0.59	0.49	0.53	0.02	0.26	0.33	0.35	0.55
		PR	0.00	0.39	0.35	0.66	0.00	0.00	0.40	0.70	0.54	0.00	0.06	0.01	0.10	0.68
		ST	0.00	0.23	0.14	0.74	0.00	0.20	0.00	0.68	0.66	0.05	0.10	0.20	0.14	0.76
OP11	TS	LI	0.00	0.16	0.00	0.43	0.00	0.26	0.36	0.75	0.63	0.05	0.35	0.31	0.33	0.52
		PR	0.00	0.23	0.14	0.73	0.00	0.09	0.06	0.75	0.65	0.06	0.11	0.16	0.18	0.76
		ST	0.04	0.48	0.24	0.67	0.10	0.52	0.18	0.72	0.70	0.19	0.49	0.37	0.48	0.89
	RS	LI	0.00	0.17	0.11	0.34	0.04	0.39	0.41	0.81	0.71	0.17	0.74	0.68	0.59	0.62
		PR	0.01	0.43	0.28	0.64	0.11	0.37	0.24	0.78	0.63	0.21	0.54	0.48	0.40	0.76
		ST	0.01	0.48	0.37	0.56	0.14	0.64	0.85	0.84	0.68	0.25	0.62	0.72	0.63	0.57
	RW	LI	0.05	0.68	0.99	0.86	0.37	0.70	0.63	0.89	0.79	0.64	0.39	0.44	0.43	0.79
		PR	0.15	0.80	0.93	0.95	0.46	0.81	0.70	0.85	0.79	0.88	0.33	0.44	0.43	0.84
		ST	0.00	0.42	0.15	0.52	0.06	0.64	0.38	0.69	0.75	0.27	0.67	0.81	0.68	0.55
OP1	TS	LI	0.07	0.60	0.72	0.95	0.37	0.74	0.14	0.87	0.83	0.66	0.52	0.55	0.50	0.82
		PR	0.23	0.60	0.62	1.00	0.38	0.78	0.18	0.69	0.77	0.79	0.46	0.46	0.48	0.83
		ST	0.03	0.41	0.28	0.45	0.25	0.68	0.51	0.81	0.89	0.43	1.00	0.94	0.81	0.65
	RS	LI	0.86	0.91	0.97	0.94	0.73	0.97	0.33	0.94	0.98	1.00	0.86	0.77	0.74	0.87
		PR	0.96	1.00	1.00	0.94	0.84	1.00	0.40	0.54	1.00	0.94	0.86	0.71	0.72	1.00
		ST	0.08	0.66	0.86	0.93	0.37	0.68	0.63	1.00	0.84	0.59	0.41	0.47	0.52	0.86
	RW	LI	0.17	0.75	0.86	0.96	0.49	0.86	0.87	0.88	0.80	0.71	0.38	0.42	0.45	0.89
		PR	0.01	0.60	0.42	0.62	0.28	0.65	0.91	0.80	0.79	0.28	0.71	0.82	0.72	0.47
		ST	0.04	0.50	0.69	0.95	0.26	0.84	0.33	0.80	0.80	0.60	0.43	0.49	0.54	0.90
OP1L	TS	LI	0.24	0.58	0.82	0.97	0.42	0.86	0.27	1.00	0.92	0.63	0.47	0.46	0.51	1.00
		PR	0.01	0.47	0.19	0.52	0.13	0.61	0.40	0.51	0.79	0.27	0.69	0.84	0.79	0.61
		ST	0.45	0.87	0.79	0.91	0.69	0.80	0.48	0.65	0.93	0.98	0.94	0.76	0.76	0.96
	RS	LI	1.00	0.93	0.97	0.97	0.64	0.95	0.54	0.71	0.84	0.93	0.84	0.73	0.73	0.97
		PR	0.08	0.51	0.21	0.46	0.45	0.76	0.54	0.64	0.81	0.45	0.99	1.00	1.00	0.54

Table 6. Normalized and averaged values of fitness function ff$(\cdot)$ for C-15 - C-28 simulation problems.

Alg.	**Sel.**	**Mut.**	**C-15**	**C-16**	**C-17**	**C-18**	**C-19**	**C-20**	**C-21**	**C-22**	**C-23**	**C-24**	**C-25**	**C-26**	**C-27**	**C-28**
		ST	0.00	0.98	0.48	0.00	0.06	0.17	0.00	0.04	0.00	0.18	0.22	0.30	0.21	0.39
	RW	LI	0.22	0.96	0.48	0.01	0.06	0.03	0.01	0.09	0.18	0.29	0.24	0.42	0.29	0.52
		PR	0.24	0.96	0.60	0.22	0.10	0.23	0.08	0.33	0.31	0.52	0.56	0.70	0.46	0.73
		ST	0.12	0.91	0.52	0.10	0.11	0.17	0.06	0.02	0.04	0.25	0.26	0.40	0.29	0.37
OPn	TS	LI	0.15	0.95	0.51	0.00	0.14	0.00	0.09	0.11	0.19	0.36	0.42	0.56	0.35	0.48
		PR	0.16	0.99	0.72	0.32	0.18	0.32	0.10	0.28	0.32	0.65	0.56	0.77	0.61	0.80
		ST	0.21	0.96	0.65	0.10	0.20	0.17	0.07	0.00	0.25	0.44	0.52	0.67	0.56	0.55
	RS	LI	0.23	0.97	0.70	0.17	0.24	0.07	0.12	0.10	0.32	0.54	0.71	0.82	0.66	0.83
		PR	0.43	1.00	0.99	0.44	0.32	0.24	0.10	0.25	0.51	0.83	0.93	1.00	0.84	1.00
		ST	0.48	0.41	0.00	0.37	0.00	0.70	0.02	0.67	0.39	0.16	0.00	0.01	0.00	0.00
	RW	LI	0.28	0.66	0.31	0.55	0.06	0.37	0.09	0.49	0.28	0.51	0.23	0.25	0.24	0.32
		PR	0.43	0.36	0.04	0.39	0.00	0.67	0.04	0.72	0.35	0.13	0.02	0.00	0.02	0.13
		ST	0.46	0.29	0.17	0.38	0.07	0.74	0.03	0.70	0.44	0.08	0.03	0.12	0.12	0.19
OP11	TS	LI	0.42	0.73	0.30	0.41	0.10	0.49	0.07	0.57	0.31	0.50	0.33	0.35	0.34	0.34
		PR	0.47	0.30	0.12	0.34	0.06	0.66	0.10	0.69	0.43	0.14	0.04	0.17	0.17	0.20
		ST	0.60	0.43	0.46	0.47	0.28	0.73	0.12	0.77	0.64	0.22	0.46	0.49	0.48	0.34
	RS	LI	0.46	0.75	0.72	0.52	0.27	0.35	0.15	0.61	0.49	0.51	0.62	0.68	0.63	0.68
		PR	0.51	0.28	0.49	0.39	0.32	0.67	0.15	0.65	0.62	0.38	0.48	0.49	0.55	0.36
		ST	0.46	0.33	0.62	0.19	0.35	0.44	0.14	0.48	0.51	0.60	0.68	0.47	0.50	0.45
	RW	LI	0.73	0.06	0.27	0.81	0.47	0.91	0.25	0.85	0.71	0.25	0.40	0.36	0.42	0.34
		PR	0.76	0.10	0.28	0.64	0.54	0.96	0.15	0.83	0.78	0.02	0.38	0.31	0.44	0.28
		ST	0.53	0.34	0.68	0.28	0.34	0.40	0.10	0.56	0.53	0.63	0.64	0.45	0.56	0.63
OP1	TS	LI	0.73	0.16	0.48	0.82	0.57	0.95	0.16	0.76	0.82	0.10	0.46	0.35	0.52	0.37
		PR	0.84	0.24	0.39	0.78	0.59	0.98	0.23	0.92	0.82	0.00	0.41	0.45	0.50	0.41
		ST	0.68	0.36	0.98	0.49	0.46	0.41	0.13	0.57	0.68	0.70	0.93	0.91	0.96	0.90
	RS	LI	0.91	0.25	0.87	0.70	0.83	0.91	1.00	0.95	0.96	0.30	0.74	0.91	0.96	0.66
		PR	1.00	0.10	0.86	1.00	0.89	0.90	0.22	0.92	1.00	0.13	0.78	0.92	0.90	0.64
		ST	0.83	0.05	0.31	0.80	0.48	0.94	0.17	0.87	0.75	0.15	0.37	0.29	0.43	0.29
	RW	LI	0.72	0.00	0.33	0.77	0.54	0.93	0.17	0.91	0.73	0.16	0.40	0.31	0.46	0.36
		PR	0.40	0.30	0.66	0.01	0.37	0.43	0.06	0.54	0.54	0.94	0.71	0.45	0.55	0.77
		ST	0.82	0.04	0.45	0.71	0.59	1.00	0.17	1.00	0.77	0.18	0.41	0.37	0.51	0.42
OP1L	TS	LI	0.87	0.05	0.42	0.75	0.64	1.00	0.13	0.95	0.78	0.09	0.40	0.41	0.52	0.30
		PR	0.57	0.41	0.76	0.05	0.39	0.35	0.19	0.59	0.54	1.00	0.67	0.57	0.64	0.79
		ST	0.85	0.09	0.81	0.69	0.91	0.94	0.26	0.98	0.99	0.14	0.78	0.97	0.90	0.62
	RS	LI	0.88	0.07	0.87	0.86	1.00	0.90	0.23	1.00	0.96	0.33	0.79	0.88	0.86	0.56
		PR	0.53	0.38	1.00	0.33	0.44	0.41	0.13	0.56	0.62	0.90	1.00	0.91	1.00	0.87

Table 7. Normalized values of fitness function $\mathrm{ff}(\cdot)$ averaged for all simulation problems.

Alg.	Sel.	Mut.	Averaged normalized minimum	Place by averaged minimum	Times best	Averaged place for each function	Average number of operators
OPn	RW	ST	0.192	1	4	7.607	2.364
		LI	0.246	3	1	9.429	2.367
		PR	0.434	13	0	16.464	2.353
	TS	ST	0.219	2	1	8.643	2.336
		LI	0.276	6	2	10.714	2.330
		PR	0.452	16	0	17.750	2.286
	RS	ST	0.331	10	3	14.143	2.350
		LI	0.404	12	0	17.036	2.334
		PR	0.619	31	0	23.393	2.280
OP11	RW	ST	0.264	5	7	10.429	2.000
		LI	0.316	9	0	9.857	2.000
		PR	0.257	4	5	8.929	2.000
	TS	ST	0.276	7	1	9.214	2.000
		LI	0.335	11	1	10.679	2.000
		PR	0.280	8	1	9.643	2.000
	RS	ST	0.448	15	0	17.964	2.000
		LI	0.472	17	0	18.929	2.000
		PR	0.436	14	0	17.536	2.000
OP1	RW	ST	0.485	19	0	19.857	1.000
		LI	0.553	22	0	21.643	1.000
		PR	0.565	26	0	21.393	1.000
	TS	ST	0.474	18	0	18.821	1.000
		LI	0.557	23	0	22.500	1.000
		PR	0.567	27	1	22.393	1.000
	RS	ST	0.617	30	0	25.036	1.000
		LI	0.815	36	0	30.214	1.000
		PR	0.792	35	0	29.214	1.000
OP1L	RW	ST	0.558	25	0	21.857	1.000
		LI	0.582	28	1	22.857	1.000
		PR	0.528	21	0	21.179	1.000
	TS	ST	0.558	24	0	22.714	1.000
		LI	0.587	29	0	23.179	1.000
		PR	0.512	20	0	21.357	1.000
	RS	ST	0.747	33	0	28.821	1.000
		LI	0.784	34	0	29.893	1.000
		PR	0.626	32	0	24.714	1.000

shows that the use of operators changes differently for different test functions, similar to OPn-RW-ST (Figure 3). At the same time, it can be seen that the requirement for each of the individuals of the population to use one operator in the OP1 caused a clear limitation on the use of some operators from the pool, although they were used in the OPn.

Table 8. Normalized values of fitness function ff$(\cdot)$ averaged for all simulation problems and grouped by selection strategies and mutation strategies of $X^{\text{op}}_{ch,1}$.

Alg.	Sel.	Avg.	Mut.	Avg.
OPn	RW	0.291	ST	0.248
	TS	0.316	LI	0.309
	RS	0.451	PR	0.502
OP11	RW	0.279	ST	0.329
	TS	0.297	LI	0.375
	RS	0.452	PR	0.324
OP1	RW	0.534	ST	0.525
	TS	0.533	LI	0.642
	RS	0.742	PR	0.641
OP1L	RW	0.556	ST	0.621
	TS	0.552	LI	0.651
	RS	0.719	PR	0.556
AVG	RW	0.415	ST	0.431
	TS	0.425	LI	0.494
	RS	0.591	PR	0.506

Table 9. Comparison of the accuracy of different PBAs based on normalized values of evaluation function ff$(\cdot)$ averaged for all simulation problems.

Alg.	Averaged normalized minimum	Place by averaged minimum
OPn-RW-ST	0.000	1
OP11-RW-PR	0.009	2
OP1-TS-ST	0.042	3
OP1L-TS-PR	0.047	4
DE	0.426	5
GA	0.767	11
GWO	0.693	9
FWA	1.000	12
PSO	0.757	10
BAT	0.685	8
CS	0.522	7
FFA	0.481	6

Table 10. A comparison of the accuracy of the best variants of the OPn, OP11, OP1, OP1L algorithms based on normalized values of fitness function ff$(\cdot)$ averaged for all simulation problems. The values in column 4 were obtained by multiplying the values from columns 2 and 3.

Alg.	Averaged normalized minimum	Averaged number of operators	Complexity versus minimum
OPn-RW-ST	0.192	2.364	0.455
OP11-RW-PR	0.257	2.000	0.513
OP1-TS-ST	0.474	1.000	0.474
OP1L-TS-PR	0.512	1.000	0.512

Table 11. A comparison of the accuracy of the average results from the OPn, OP11, OP1, OP1L algorithms based on normalized values of fitness function ff$(\cdot)$ averaged for all simulation problems. The values in column 4 were obtained by multiplying the values from columns 2 and 3.

Alg.	Averaged normalized minimum	Averaged number of operators	Complexity versus minimum
OPn	0.353	2.333	0.823
OP11	0.343	2.000	0.685
OP1	0.603	1.000	0.603
OP1L	0.609	1.000	0.609

3.3 Comparison of OP1 with other PBAs

A comparison of the main features of the selected PBAs is provided in Table 1. It shows that the family of the OPn, OP11 and OP1 algorithms gives a new look at the algorithm design process and has several important advantages in the PBAs group. A direct comparison of the accuracy of PBAs is difficult because the authors use different algorithms and parameters of the evolution process. Therefore, to compare the OP1 algorithm with other PBAs, we implemented selected methods and tested them in accordance with the information provided in Section 3.1. These methods include: Operator-based Population n (OPn) [27], Operator-based Population 1+1 (OP11) [29], Differential Evolution (DE) [41], Genetic Algorithm (GA) [11], Grey Wolf Optimizer (GWO) [31], Firework Algorithm (FWA) [43], Particle Swarm Optimization (PSO) [23], Bat Algorithm (BAT) [51], Cuckoo Search (CS) [50], and Firefly Algorithm (FFA) [49].

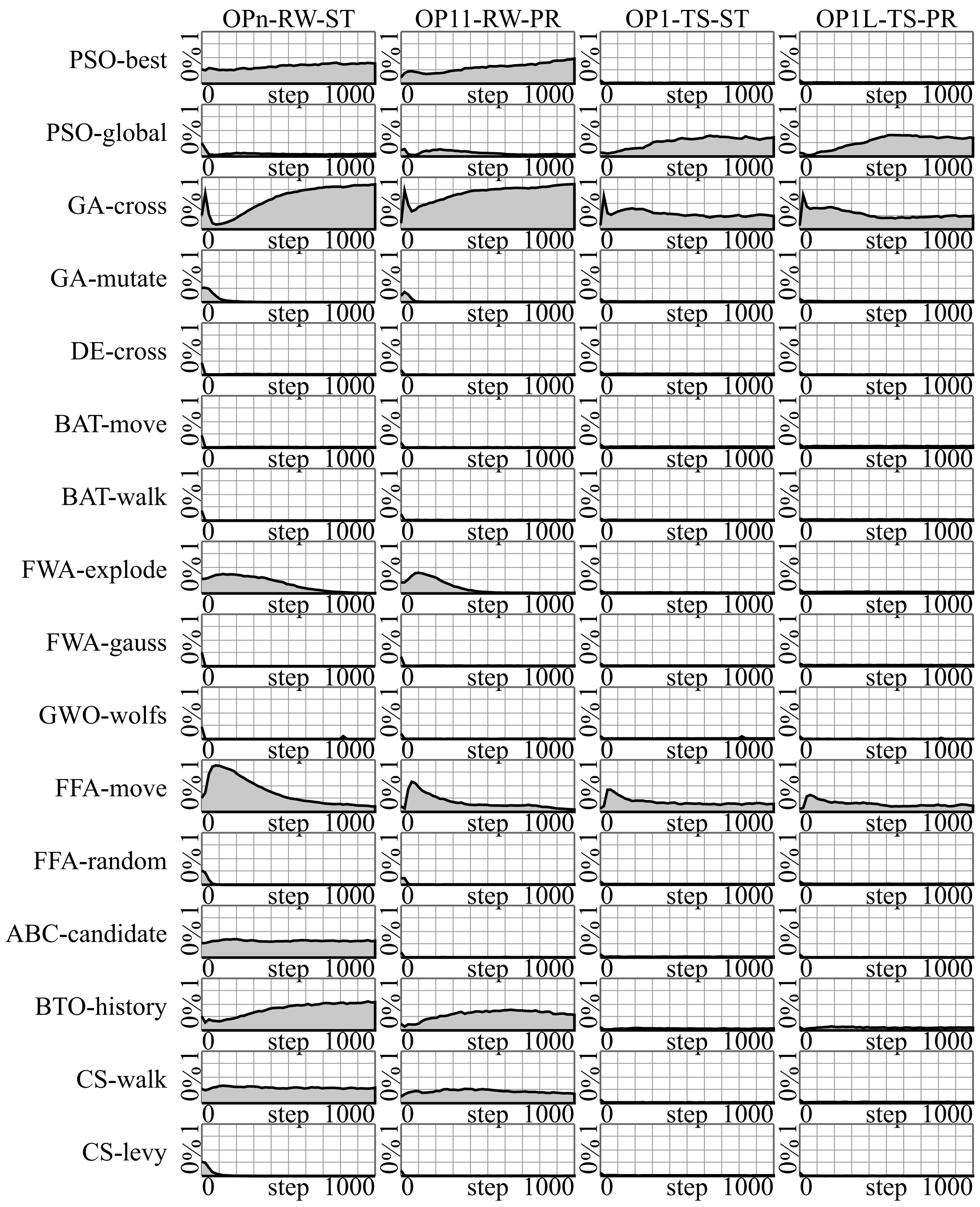


Figure 2. Comparison of the average operator use by the best PBAs variants (see Table 10) for test function C-10.

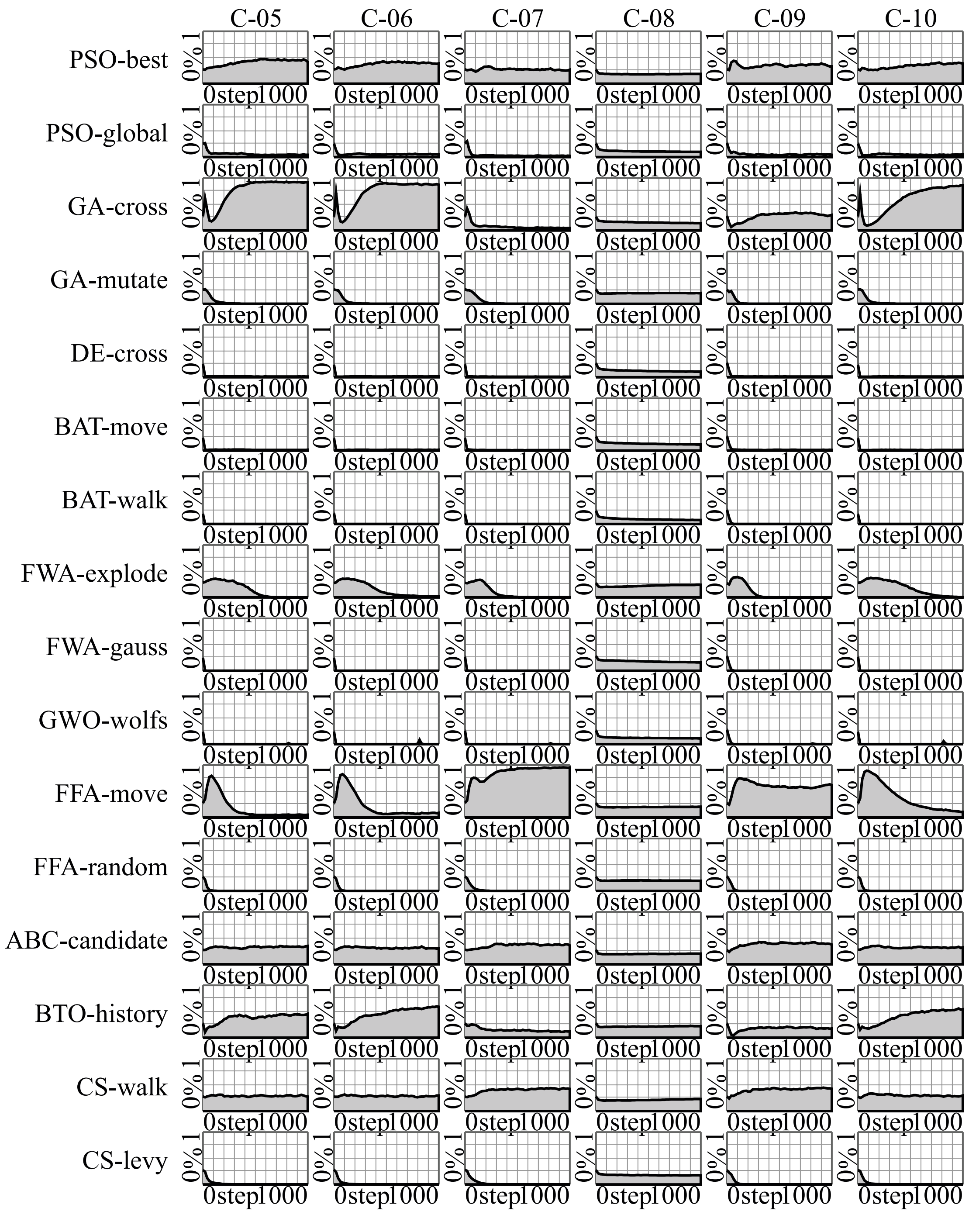


Figure 3. Comparison of the average operator usage for test functions C-05 - C-10 and the OPn-RW-ST algorithm.

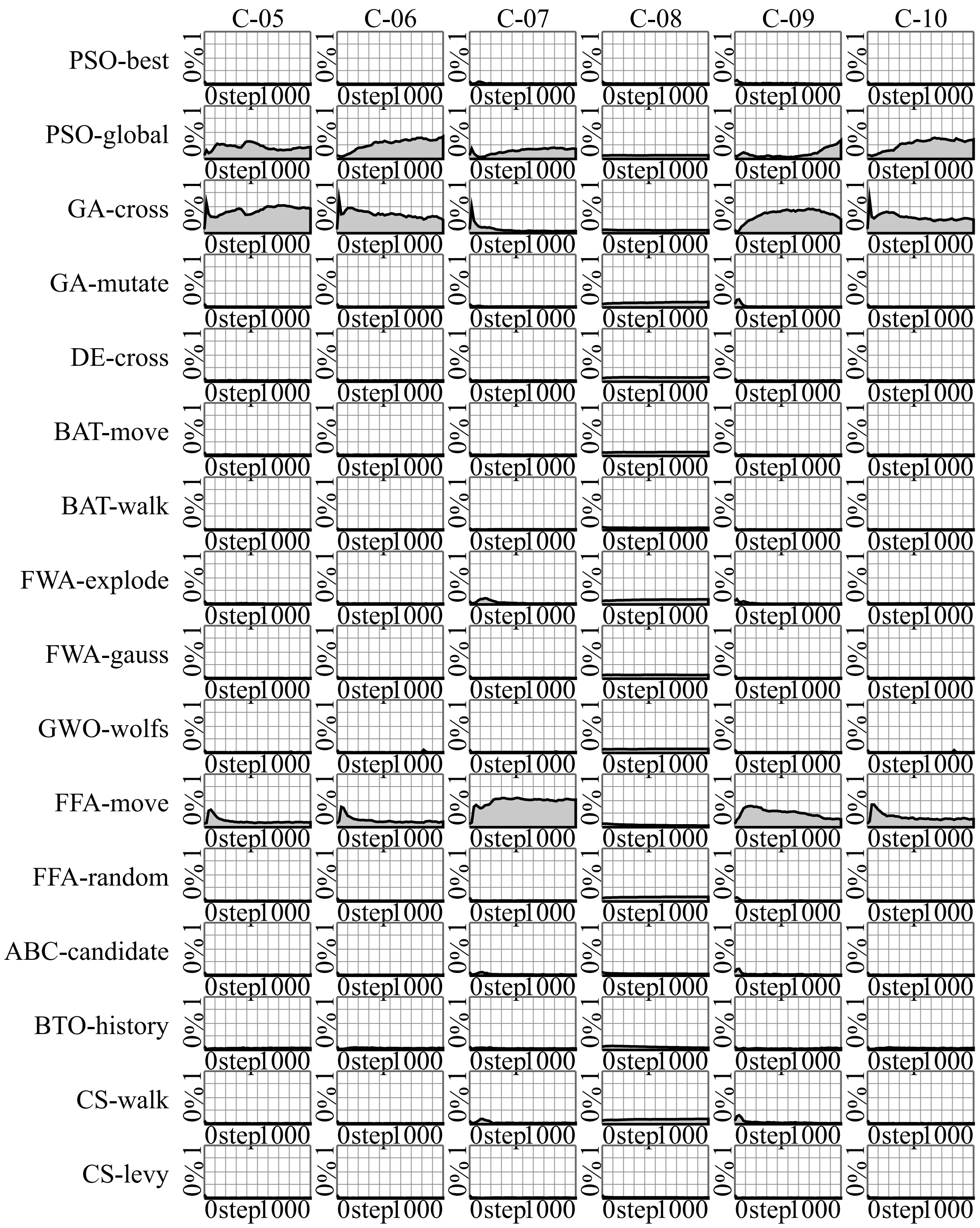


Figure 4. Comparison of the average operator usage for test functions C-05 - C-10 and the OP1-TS-ST algorithm.

The conclusions from the comparison can be summarized as follows:

– Comparison of the accuracy of different PBAs based on normalized values of fitness function $\mathrm{ff}(\cdot)$ averaged for all simulation problems is shown in Table 9. The results obtained for the OPn, OP11, OP1, and OP1L algorithms are an order of magnitude better than those obtained for other PBAs (Table 9). Therefore, the approach to change the search operators during the evolution process seems very effective.

– Comparison of the accuracy of the best variants of the OPn, OP11, OP1, OP1L algorithms based on normalized values of the fitness function is shown in Table 10. The best results and the best ratio of complexity to accuracy were obtained by the OPn-RW-ST (Table 10, columns 2 and 4). The OP1 and OP1L methods obtained acceptable results, but slightly worse than those obtained by the OPn and OP11.

– Comparison of the accuracy of average algorithms OPn, OP11, OP1, OP1L based on normalized values of the evaluation function is shown in Table 11. The best results were obtained for the OP11 algorithm (Table 11, column 2). However, the best ratio of complexity to accuracy was obtained for the OP1 algorithm (Table 11, column 4). Therefore, when the key criterion is the complexity of the algorithm, then the OP1 method which generates acceptable solutions with minimal use of the operator pool can be used. If this is accuracy that is expected the key criterion, then the OP11 method can be used.

4 Conclusions

In this paper, a new population-based algorithm (PBA) is proposed: an evolutionary algorithm with an automatic selection of the search mechanism (OP1). Its most important feature is that for each individual of the population, the evolutionary operator is individually selected. The exchange of operators in a population is thus a dynamic process.

The OP1 algorithm was tested using 28 considered known simulation problems C-01 - C-28. For the OP1 slightly worse results than in the case of the OPn and OP11 were obtained, but these are satisfactory and obtained with minimal use of the operator pool. At the same time, these results are significantly better than those obtained for popular PBAs that did not use the operator exchange mechanism.

The mutation of operators with protective thresholds found in the OP1L variant did not give as good results as expected, but they are acceptable and thus worth further investigation.

Our plans include developing a multi-population and multi-criteria version of the proposed algorithm.

Acknowledgment

The authors would like to thank the associate editor and reviewers for their helpful comments.

This paper was financed under the program of the Minister of Science and Higher Education under the name 'Regional Initiative of Excellence' in the years 2019-2022, project number 020/RID/2018/19 with the amount of financing PLN 12 000 000.

References

[1] S.P. Adam, S.A.N. Alexandropoulos, P.M. Pardalos, M.N. Vrahatis, No free lunch theorem: a review, Approximation and Optimization, Springer, 57-82, 2019.

[2] E.S. Ali, S.M. Abd-Elazim, Bacteria foraging optimization algorithm based load frequency controller for interconnected power system, Int. J. of Electrical Power & Energy Systems, 33(3), 633-638, 2011.

[3] T. de Fátima Araújo, W. Uturbey, Performance assessment of PSO, DE and hybrid PSO–DE algorithms when applied to the dispatch of generation and demand, Int. J. of Electrical Power & Energy Systems, 47, 205-217, 2013.

[4] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, 2007 IEEE Congress on Evolutionary Comp., 2007.

[5] Ł. Bartczuk, A. Przybył, K. Cpałka, A new approach to nonlinear modelling of dynamic systems based on fuzzy rules, Int. J. of Applied Mathematics and Computer Science, 26(3), 603-621, 2016.

[6] Z.S. Chen, B. Zhu, Y.L. He, L.A. Yu, A PSO based virtual sample generation method for small sample

sets: Applications to regression datasets, Engineering Applications of Artificial Intelligence, 59, 236-243, 2017.

[7] S. Chu, P. Tsai, J. Pan, Cat Swarm Optimization, LNCS, 4099, 854-858, 2006.

[8] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Applied Mathematics and Computation, 219(15), 8121–8144, 2013.

[9] K. Cpałka, Design of interpretable fuzzy systems, Springer, 2017.

[10] M. Črepinšek, S. H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM computing surveys (CSUR), 45(3), 1-33, 2013.

[11] L. Davis, Handbook of genetic algorithms, 1991.

[12] D. Dawar & S.A. Ludwig, Effect of Strategy Adaptation on Differential Evolution in Presence and Absence of Parameter Adaptation: An Investigation, J. of Artificial Intelligence and Soft Computing Research, 8(3), 211-235, 2018.

[13] J. Del Ser, E. Osaba, D. Molina, X.S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: Where we stand and what's next, Swarm and Evolutionary Computation, 48, 220-250, 2019.

[14] H. Faris, I. Aljarah, M.A. Al-Betar, S. Mirjalili, Grey wolf optimizer: a review of recent variants and applications, Neural Computing and Applications, 30(2), 2018, 413-435.

[15] A.H. Gandomi, X.S. Yang, S. Talatahari, S. Deb, Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization, Computers & Mathematics with Applications, 63(1), 191-200, 2012.

[16] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Applied Mathematics and Computation, 274, 292-305, 2016.

[17] Z.W. Geem, J.H. Kim, G. Loganathan, A New Heuristic Optimization Algorithm: Harmony Search, Simulation, 76(2), 60-68, 2001.

[18] E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., 1989.

[19] D. Grochol, L. Sekanina, M. Zadnik, J. Korenek, V. Kosar, Evolutionary circuit design for fast FPGA-based classification of network application protocols, Applied Soft Computing, 38, 933-941, 2016.

[20] K. Hussain, M.N.M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artificial Intelligence Review, 52(4), 2191-2233, 2019.

[21] T. Jayabarathi, T. Raghunathan, B.R. Adarsh, P.N. Suganthan, Economic dispatch using hybrid grey wolf optimizer, Energy, 111, 630-641, 2016.

[22] D. Karaboga, B. Basturk, 2007, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, LNAI, 4529, Berlin:Springer-Verlag, 789–98, 2007.

[23] J. Kennedy, Particle swarm optimization, Encyclopedia of Machine Learning, 760-766, 2010.

[24] J. Kennedy, R. Eberhart, Particle swarm optimization, Proc. IEEE Int. Conf. on Neural Networks, 4, 1942-1948, 1995.

[25] E. Krell, A. Sheta, A.P.R. Balasubramanian, S.A. King, Collision-Free Autonomous Robot Navigation in Unknown Environments Utilizing PSO for Path Planning. J. of Artificial Intelligence and Soft Computing Research, 9(4), 267-282, 2019.

[26] K. Łapa, Meta-optimization of multi-objective population-based algorithms using multi-objective performance metrics, Information Sciences, 489, 193-204, 2019.

[27] K. Łapa, K. Cpałka, Flexible fuzzy PID controller (FFPIDC) and a nature-inspired method for its construction, IEEE Trans. on Industrial Informatics, 14(3), 1078-1088, 2018.

[28] K. Łapa, K. Cpałka, L. Wang, New method for design of fuzzy systems for nonlinear modelling using different criteria of interpretability, Artificial Intelligence and Soft Computing, LNCS, 8467, Springer, 217-232, 2014.

[29] K. Łapa, K. Cpałka, M. Zalasiński, Algorithm Based on Population with a Flexible Search Mechanism, IEEE Access, 7, 132253-132270, 2019.

[30] J. Luo, J. Liu, Y. Hu, An MILP model and a hybrid evolutionary algorithm for integrated operation optimisation of multi-head surface mounting machines in PCB assembly, Int. J. of Production Research, 55(1), 145-160, 2017.

[31] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in Engineering Software, 69, 46-61, 2014.

[32] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in Engineering Software, 95, 51-67, 2016.

[33] M. Mizera, P. Nowotarski, A. Byrski, M. Kisiel-Dorohinicki, Fine Tuning of Agent-Based Evolutionary Computing, J. of Artificial Intelligence and Soft Computing Research, 9(2), 81-97, 2019.

[34] K. Ono, Y. Hanada, M. Kumano, M. Kimura, Enhancing Island Model Genetic Programming by Controlling Frequent Trees, J. of Artificial Intelligence and Soft Computing Research, 9(1), 51-65, 2019.

[35] E. Osaba, F. Diaz, E. Onieva, Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts, Applied Intelligence, 41(1), 145-166, 2014.

[36] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A Gravitational Search Algorithm, Information Sciences, 179(13), 2232-2248, 2009.

[37] L. Rutkowski, Identification of MISO nonlinear regressions in the presence of a wide class of disturbances. IEEE Trans. on Information Theory, 37(1), 214-216, 1991.

[38] L. Rutkowski, Computational intelligence: methods and techniques, Springer Science & Business Media, 2008.

[39] S. Sadiqbatcha, S. Jafarzadeh, Y. Ampatzidis, Particle Swarm Optimization for Solving a Class of Type-1 And Type-2 Fuzzy Nonlinear Equations, J. of Artificial Intelligence and Soft Computing Research, 8(2), 103-110, 2018.

[40] K. Sörensen, Metaheuristics—the metaphor exposed, Int. Trans. in Operational Research, 22(1), 3-18, 2015.

[41] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. of Global Optimization, 11(4), 341-359, 1997.

[42] J. Szczypta, A. Przybył, K. Cpałka, Some aspects of evolutionary designing optimal controllers, Artificial Intelligence and Soft Computing, LNCS, 7895, Springer, 91-100, 2013.

[43] Y. Tan, Y. Zhu, Fireworks Algorithm for Optimization, LNCS, 6145, 355-364, 2010.

[44] G. Tambouratzis, Using particle swarm optimization to accurately identify syntactic phrases in free text. J. of Artificial Intelligence and Soft Computing Research, 8(1), 63-77, 2018.

[45] D. Teodorovic, P. Lucic, G. Markovic, M. D. Orco, Bee Colony Optimization: Principles and Applications, 2006 8th Seminar on Neural Network Applications in Electrical Engineering, 2006.

[46] B. Wang, X. Jin, B. Cheng, Lion pride optimizer: An optimization algorithm inspired by lion pride behavior, Science China Information Sciences, 55(10), 2369-2389, 2012.

[47] J.J. Liang, B.Y. Qu, P.N. Suganthan, A.G. Hernandez-Diaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, https://al-roomi.org/multimedia/ CEC_Database/ CEC2013/ RealParameterOptimization/ _TechnicalReport.pdf, 2013.

[48] Y. Xu, O. Ding, R. Qu, K. Li, Hybrid multi-objective evolutionary algorithms based on decomposition for wireless sensor network coverage optimization, Applied Soft Computing, 68, 268-282, 2018.

[49] X. Yang, Firefly Algorithms for Multimodal Optimization, Stochastic Algorithms: Foundations and Applications, 169-178, 2009.

[50] X. Yang, S. Deb, Cuckoo Search via Levy flights, 2009 World Congress on Nature & Biologically Inspired Computing, 2009.

[51] X. Yang, A new metaheuristic bat-inspired algorithm, Nature inspired cooperative strategies for optimization, 65-74, 2010.

[52] M. Zalasiński, K. Cpałka, New algorithm for on-line signature verification using characteristic hybrid partitions, Information Systems Architecture and Technology: Proc. of 36th Int. Conf. on Information Systems Architecture and Technology – ISAT 2015 – Part IV, Advances in Intelligent Systems and Computing, 432, Springer, 147-157, 2016.

[53] M. Zalasiński, K. Cpałka, Novel algorithm for the on-line signature verification using selected discretization points groups, Artificial Intelligence and Soft Computing, LNCS, 7894, Springer, 493-502, 2013.

[54] M. Zalasiński, K. Cpałka, Y. Hayashi, New fast algorithm for the dynamic signature verification using global features values, Artificial Intelligence and Soft Computing, LNCS, 9120, Springer, 175-188, 2015.

[55] M. Zalasiński, K. Cpałka, E. Rakus-Andersson, An idea of the dynamic signature verification based on a hybrid approach, Artificial Intelligence and Soft Computing, LNCS, 9693, Springer, 232-246, 2016.

[56] M. Zalasiński, K. Łapa, K. Cpałka, Prediction of values of the dynamic signature features, Expert Systems with Applications, 104, 86-96, 2018.

Krystian Łapa received the M.Sc. and Ph.D. degrees from the Częstochowa University of Technology, Częstochowa, Poland, in 2010 and 2015, respectively. He is currently an Associate Professor with the Department of Computer Engineering. Dr. Łapa has authored over 40 publications. His current research interests include computational intelligence, nature-inspired methods, and expert systems.

Krzysztof Cpałka was born in Częstochowa, Poland, in 1972. received the M.Sc. and Ph.D. degrees from the Częstochowa University of Technology, Częstochowa, Poland, in 1997 and 2002, respectively, and the D.Sc. degree in Computer Science from the Systems Research Institute of the Polish Academy of Sciences in Warsaw, Warsaw, Poland, in 2010. Since 2010, he has been a professor with the Department of Computer Engineering, Częstochowa University of Technology. He is the author or co-author of two books and more than 100 papers, including several papers in various series of IEEE Transactions. He was a recipient of the IEEE Transactions on Neural Networks Outstanding Paper Award in 2005. His research interests include soft computing, computational intelligence, machine learning, and bio-inspired global optimization algorithms and their applications.

Łukasz Laskowski received Ms.C. and Ph.D. degrees from the Częstochowa University of Technology, Częstochowa, Poland, in 2004 and 2009, respectively. He is currently an Associate Professor with the Department of Molecular Engineering and Nanoelectronics at Institut of Nuclear Physics of the Polish Academy of Sciences. Dr. Laskowski has authored over 30 publications. His current research interests include artificial neural networks, nanoelectronics, molecular engineering, and nanotechnology.

Andrzej Cader is a professor at University of Social Science in Łódź, Poland. He received the Ph.D. degree in biocybernetics and biomedical engineering from the Medical University in Łódź, Poland. His research interests include intelligent systems science based on several architectures such as neural networks and complex systems. He is also currently engaged in time series analysis and chaos theory.

Zhigang Zeng received his B.S. degree from Hubei Normal University, Huangshi, China, and his M.S. degree from Hubei University, Wuhan, China, in 1993 and 1996, respectively, and his Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2003. He is a professor in School of Automation, Huazhong University of Science and Technology, Wuhan, China, and also in the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, China. His current research interests include neural networks, switched systems, computational intelligence, stability analysis of dynamic systems, pattern recognition and associative memories.