Applied Mathematics and Computation 347 (2019) 865–881

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Novelty search for global optimization

Iztok Fister[a,b], Andres Iglesias[b], Akemi Galvez[b], Javier Del Ser[e,d,c], Eneko Osaba[e], Iztok Fister Jr.[a], Matjaž Perc[f,g,h,*], Mitja Slavinec[f]

[a] Faculty of Electrical Engineering and Computer Science, University of Maribor,Maribor SI-2000, Slovenia
[b] University of Cantabria, Avenida de los Castros, s/n, Santander 39005, Spain
[c] University of the Basque Country (UPV/EHU), Bilbao, Spain
[d] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain
[e] TECNALIA Research and Innovation, 48160 Derio, Spain
[f] Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, Maribor SI-2000, Slovenia
[g] CAMTP–Center for Applied Mathematics and Theoretical Physics, University of Maribor, Mladinska 3, Maribor SI-2000, Slovenia
[h] Complexity Science Hub Vienna, Josefstädterstraße 39, Vienna A-1080, Austria

A R T I C L E I N F O

Keywords:
Novelty search
Differential evolution
Swarm intelligence
Evolutionary robotics
Artificial life

A B S T R A C T

Novelty search is a tool in evolutionary and swarm robotics for maintaining the diversity of population needed for continuous robotic operation. It enables nature-inspired algorithms to evaluate solutions on the basis of the distance to their *k*-nearest neighbors in the search space. Besides this, the fitness function represents an additional measure for evaluating the solution, with the purpose of preserving the so-named novelty solutions into the next generation. In this study, a differential evolution was hybridized with novelty search. The differential evolution is a well-known algorithm for global optimization, which is applied to improve the results obtained by the other solvers on the CEC-14 benchmark function suite. Furthermore, functions of different dimensions were taken into consideration, and the influence of the various novelty search parameters was analyzed. The results of experiments show a great potential for using novelty search in global optimization.

1. Introduction

Successful applications of Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) based algorithms to almost every domain of human activity have proven that the so-named family of nature-inspired algorithms has reached their maturity phase, and, therefore, calls for new challenges. Several challenges have emerged with the development of Evolutionary Robotics (ER) and Swarm Robotics (SR), where the nature-inspired algorithms are embedded into hardware [9]. Together with the hardware, these algorithms act as autonomous artificial agents (robots) that are embodied into an environment to solve a certain task, which can be mathematically formulated as an optimization problem. In this case, the quality of the generated solutions cannot be evaluated in the phenotype space directly, but their estimation must be obtained after observing how well the phenotype reacts in the conditions of the dynamic process governed by the environment. Therefore, the traditional three-step fitness function evaluation chain, consisting of genotype-phenotype-fitness in Evolutionary Com-

[*] Corresponding author at: Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, Maribor SI-2000, Slovenia.
E-mail addresses: iztok.fister@um.si (I. Fister), iglesias@unican.es (A. Iglesias), galveza@unican.es (A. Galvez), javier.delser@tecnalia.com (J. Del Ser), eneko.osaba@tecnalia.com (E. Osaba), iztok.fister1@um.si (I. Fister Jr.), mitja.slavinec@um.si (M. Slavinec).

https://doi.org/10.1016/j.amc.2018.11.052
0096-3003/© 2018 Elsevier Inc. All rights reserved.

putation (EC), has been replaced by a four-step evaluation chain genotype-phenotype-behavior-fitness in nature-inspired robotics that comprise the ER and SR.

However, the implementation of the nature-inspired algorithms into hardware, and the rapid development of nature-inspired robotics has introduced new problems [9]. Selection pressure is one of the more serious issues, and directs the search process towards the best solutions at the expense of losing the population diversity. Usually, the loss of population diversity causes that the search process gets stuck into local optima. As a result, this phenomenon causes the evolution process to terminate prematurely. Unfortunately, this is in contrast with the demands of an open-ended evolution [22] that are a prerequisite for regular readiness of nature-inspired robots. In fact, these demands are subjects of Artificial Life (ALife) that discover conditions under which the open-ended evolution can occur [25].

Multi-Objective Evolutionary Algorithms (MOEAs) [4] represent the biggest advantage of the nature-inspired algorithms, enabling the nature-inspired robotic community to tackle the selection pressure successfully [8] by evaluating each solution according to more than one objective. This advantage is also exploited by Novelty Search (NS) that guides the evolutionary search, using not only one fitness function, but also evaluates each solution according to its novelty. On the other hand, there are two components of the evolutionary search process that have a huge effect on the population diversity, i.e., exploration and exploitation. Črepinšek et al. in [30] expose three issues referring to these components: (1) Which components of EAs contribute to the exploration/exploitation, (2) how to control, and (3) how to balance the exploration/exploitation in the evolutionary search process. In the sense of this study, NS participates by overcoming the last two issues.

NS was introduced by Lehman and Stanley [18]; it was developed on the basis of acknowledging that the natural selection does not always explain the increases in evolutionary complexity [27]. Usually, fitness function does not reward the intermediate stepping stones leading the search process to the optimal solutions. Therefore, NS substitutes the fitness function with the novelty measure [7], which gauges the distance of the novelty solutions to their k-th nearest neighbors in a search space. Moreover, NS prevents a deception of objective functions and premature convergence [16]. In that case, it is focused on direct search for novelty individuals, and does not wait for the search process to discover them on the fly. Since 2008, NS has been applied for solving a wide range of problems in different domains, like robot controllers [19], multirobot systems [15], machine learning [24], and games [21].

Interestingly, the open-ended paradigm is not the only one that does not completely agree with Darwinian evolution [3]. For instance, the neutral theory of molecular evolution of Kimura [17] argues that the majority of mutations are neutral at molecular level. Therefore, the evolutionary process is guided by genetic drift that can cause losing some genetic traits, especially in smaller isolated populations. This theory represented an inspiration for the emergence of the so-called neutral selection operator proposed in [12], where the most distant solution from the mass center of a set of neutral solutions is selected (i.e., solutions with the same fitness).

This paper is actually an extension of the conference paper by Fister et al. [11], which was applied for solving the CEC-2014 function benchmark suite, where functions of dimension $D = 10$ were observed. On the other hand, the best parameter setting for NS was searched using exhaustive search, where all combination of parameters were enumerated. The purpose of the mentioned paper is to verify that NS can be applied successfully within nature-inspired algorithms. To the best of our knowledge, few effort is invested into using it for solving the global optimization problems. Although NS could be applied to any nature-inspired algorithm, we select a differential evolution (DE) [28] due to its global search abilities. In line with this, two versions of the algorithm are hybridized with NS as follows:

- The original DE [28], denoted as nDE;
- The self-adaptive jDE [1], denoted as njDE.

Although the NS was primarily applied in evolutionary robotics, both the proposed algorithms were not embodied into hardware, but rather applied for solving the global optimization problems on a disembodied computer system. In this case, performing a search for novelties is not possible in the behavior space, but in the phenotype space. Indeed, many points in a phenotype space collapse to the same points in the behavior space. This means that the search for novelties can be performed in a search space independently of environment conditions [18]. When we assume that the points in the phenotype space are collapsed with points in the behavior space, running these algorithms on the disembodied computer system is exactly the same as running on the embodied hardware.

All algorithms used in our study were applied for solving the CEC-2014 benchmark suite. Thus, our motivation was to show that the new DE algorithms, hybridized with NS, maintain higher population diversity, and, therefore, can improve the results of their original counterparts. In order to justify this assertion, different dimensions of the testing function were taken into consideration. Moreover, the NS is defined using various parameters very loosely, which demands that extensive adjustment is needed before using it. This analysis highlights how the parameter settings influence the results of the optimization. Furthermore, the results of the hybrid algorithms were compared with other state-of-the-art algorithms, like L-Shade [29] (the winner of the CEC-2014 Competition on Real-Parameter Single Objective) and MVMO [10], in order to show that they could also be comparable with these. Finally, the obtained results are discussed in detail.

The remainder of the paper is structured as follows. Section 2 refers to highlighting the background information. A description of different DE variants hybridized with NS is presented in Section 3. The results of experiments are illustrated and discussed in Section 4. Summarizing the performed work and outlining possible directions of the further work are the subjects of the last section.

2. Background material

This section is devoted to presenting the background information needed for understanding the subjects that follow. In line with this, the following subjects are taken into consideration:

- Differential evolution,
- the self-adaptive Differential evolution,
- Novelty search.

The mentioned subjects are discussed in the remainder of the section.

2.1. Differential evolution

DE belongs to the class of stochastic nature-inspired population-based algorithms and is appropriate for solving continuous, as well as discrete, optimization problems. DE was introduced by Storn and Price in 1997 [28] and, since then, many DE variants have been proposed. The original DE algorithm is represented by real-valued vectors:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}), \quad \text{for } i = 1, \ldots, NP,$$

on which operators are applied, such as mutation, crossover, and selection.

In the basic DE mutation, two solutions are selected randomly and their scaled difference is added to the third solution, as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r0}^{(t)} + F.(\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}), \tag{1}$$

where $F \in [0.1, 1.0]$ denotes the scaling factor that scales the magnitude of modification, while NP represents the population size and $r0, r1, r2$ are randomly selected values in the set $\{1, \ldots, NP\}$. Note that the proposed interval of values for parameter F was enforced in the DE community, although Price and Storn originally proposed the slightly different interval, i.e., $F \in [0.0, 2.0]$.

The trial vector is built from parameter values copied from either i-th mutant vector generated by Eq. (1) or corresponding i-th parent vector. Mathematically, the crossover can be expressed as follows:

$$w_{i,j}^{(t)} = \begin{cases} u_{i,j}^{(t)}, & \text{rand}_j(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(t)}, & \text{otherwise}, \end{cases} \tag{2}$$

where $\text{rand}_j(0, 1)$ denotes the random value drawn from uniform distribution in interval [0,1], and $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution. The condition $j = j_{rand}$, where the j_{rand} is the randomly selected position in the trial vector, ensures that this differs from the original solution $\mathbf{x}_i^{(t)}$ in at least one element.

Mathematically, the selection can be expressed as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)}, & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)}, & \text{otherwise}. \end{cases} \tag{3}$$

The selection is usually called 'one-to-one', because trial and the corresponding i-th parent vector compete for surviving in the next generation. However, the best among them according to the fitness function will survive.

Crossover and mutation can be performed in several ways in DE. Therefore, a specific notation was introduced to describe the varieties of these methods (also strategies), in general. For example, 'rand/1/bin' presented in Eq. (1) denotes that the base vector is selected randomly, one vector difference is added to it, and the number of modified parameters in the trial vector follows a binomial distribution.

2.2. jDE algorithm

In 2006, Brest et al. [1] proposed an efficient DE variant (jDE), where control parameters are self-adapted during the search process. In this case, two parameters, namely scale factor F and crossover rate CR, are added to the representation of every individual, and undergo operation of the variation operators. As a result, the individual in jDE is represented as follows:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}, F_i^{(t)}, CR_i^{(t)}), \quad \text{for } i = 1, \ldots, NP.$$

The jDE modifies parameters F and CR according to the following equations:

$$F_i^{(t+1)} = \begin{cases} F_l + \text{rand}_1 * (F_u - F_l) & \text{if } \text{rand}_2 < \tau_1, \\ F_i^{(t)} & \text{otherwise}, \end{cases} \tag{4}$$

$$CR_i^{(t+1)} = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_4 < \tau_2, \\ CR_i^{(t)} & \text{otherwise}, \end{cases} \tag{5}$$

where: $\text{rand}_{i=1,\ldots,4} \in [0, 1]$ are randomly generated values drawn from uniform distribution in the interval [0,1], τ_1 and τ_2 are learning steps, and F_l and F_u are the lower and upper bounds for parameter F, respectively.

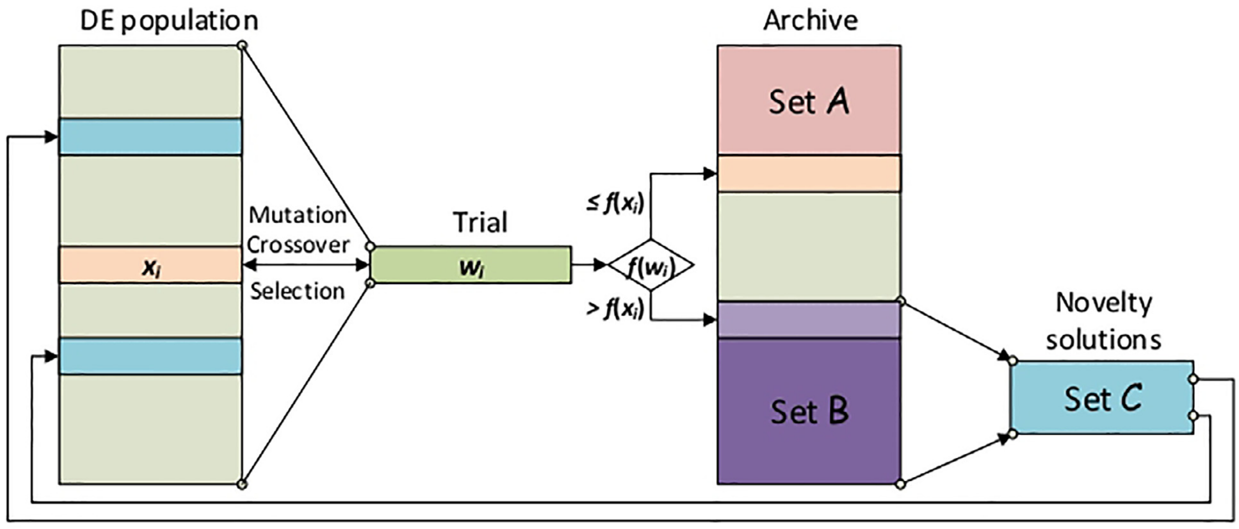

Fig. 1. The new DE population model.

2.3. Novelty search

Originally, NS measures the mean distance between an individual $\mathbf{x}_i$ from the population of solutions and k-th nearest neighbors in a behavior space $\mathcal{N}(\mathbf{x}_i) = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, in other words [18]:

$$\rho(\mathbf{x}_i) = \frac{1}{k}\sum_{j=1}^{k} dist(\mathbf{x}_i, \boldsymbol{\mu}_j), \quad \text{and } \mathbf{x}_i \neq \boldsymbol{\mu}_j, \tag{6}$$

where the k-th nearest neighbor is determined with respect to the behavior distance metric *dist*. The parameter k (neighborhood size) is problem-dependent, and must be determined experimentally. The same is true also for the distance metric, which must be selected according to the characteristics of the problem to be solved.

In summary, NS is loosely defined and leaves to the developer's imagination the decision how to tailor the search so that the results will be as good as possible [7].

3. Proposed novelty differential evolution

Indeed, introducing NS into DE/jDE requires the evaluation of solutions according to two conflicting objectives: (1) the problem-dependent (i.e., fitness function), and (2) the population-diversity objective (NS). On the other hand, this modification changes the way of problem handling. Instead of dealing with a problem as a single-objective, this is now treated as multi-objective. However, NS needs to be adjusted, solving the problem of interest first.

Then, the DE population model is modified as illustrated in Fig. 1. The new DE population model acts as follows. Parallel to the traditional DE 'one-to-one' selection, where a trial solution $\mathbf{w}_i$, obtained with the proper DE mutation strategy and DE crossover, competes with corresponding original solution $\mathbf{x}_i$ for surviving into the next generation, an archive of trial vectors is created. Trial vectors are archived in two ways. Firstly, when the better trial solution is found, then the parent is normally replaced in the DE population, while the trial is simultaneously stored on the first free position from top to bottom of the archive. Secondly, when the trial solution is worse, this is not eliminated, but stored in the first free position from bottom to top of the archive. Finally, the archive is divided into two parts, i.e., the higher, denoted as set $\mathcal{A}$, contains solutions which survived due to satisfying the first objective and the lower, denoted as set $\mathcal{B}$, consists of eliminated solutions that do not satisfy the first objective.

Actually, the eliminated solutions in set $\mathcal{B}$ present the potential novelty solutions, from which the set of novelty solutions $\mathcal{C}$ are selected according to the second objective. For each potential novelty solution $\mathbf{x}_i \in \mathcal{B}$, its k-th nearest neighbors $\mathcal{N}(\mathbf{x}_i) = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ are selected from the set of survivor solutions $\boldsymbol{\mu}_j \in \mathcal{A}$ according to a behavior distance metric *dist*. The behavior distance metric is calculated according to:

$$dist(\mathbf{x}_i, \boldsymbol{\mu}_j) = \begin{cases} \frac{d(\mathbf{x}_i, \boldsymbol{\mu}_j)}{\sigma_{sh}}, & d(\mathbf{x}_i, \boldsymbol{\mu}_j) > \sigma_{sh}, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$ denotes the Euclidean distance between vectors $\mathbf{x}_i$ and $\boldsymbol{\mu}_j$, and the novelty area width σ_{sh} determines the distance needed for recognizing the novelty solution. Actually, the best k solutions satisfying Eq. (7) are included into the

k-th nearest neighborhood of vector $\mathbf{x}_i$, for which a NS metric $\rho(\mathbf{x}_i)$ is calculated according to Eq. (6). All the solutions in set $\mathcal{B}$ are then sorted in descending order according to the values of the NS metric. Finally, the first R solutions are included into the $\mathcal{C}$ set of novelty solutions.

The pseudo-code of the NS algorithm is illustrated in Algorithm 1.

Algorithm 1 NC within the original DE algorithm.

```
1: procedure NOVELTY SEARCH
2:     $\mathcal{A} = \{\exists \boldsymbol{\mu}_i : f(\mathbf{w}_i) \leq f(\mathbf{x}_j) \wedge i \neq j\}$;   // set of survivor solutions
3:     $\mathcal{B} = \{\exists \mathbf{x}_i : f(\mathbf{w}_i) > f(\mathbf{x}_j) \wedge i \neq j\}$;   // set of eliminated solutions
4:     if $|\mathcal{A}| < k$ then   // number of survivor solutions less than the neighborhood?
5:         $\mathcal{A} = \mathcal{A} \cup \mathcal{B}$;   ~// increases the neighborhood set to the whole population
6:     end if
7:     $\forall \mathbf{x}_i \in \mathcal{B} : \exists \mathcal{N}(\mathbf{x}_i) : \boldsymbol{\mu}_j \in \mathcal{A} \wedge \mathbf{x}_i \neq \boldsymbol{\mu}_j \wedge |\mathcal{N}(\mathbf{x}_i)| \leq k$; // select $k$-nearest neighbors
8:     $\forall \mathbf{x}_i \in \mathcal{B} : \rho(\mathbf{x}_i)$;                // calculate their novelty values
9:     $\mathcal{C} = \{\forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{B} : \max |\rho(\mathbf{x}_i) - \rho(\mathbf{x}_j| \wedge i \neq j \wedge |\mathcal{C}| \leq R\}$;
10: end procedure
```

It is important to notice that NS is launched as the last operator in the traditional DE evolution cycle and, thus, affects the complexity of the original DE substantially. As a matter of fact, the nDE/njDE introduces two new parameters:

- Replacement size $R \in [1, Np]$ that limits the number of novelty solutions,
- Learning rate parameter $\tau_3 \in [0, 1]$ that controls how often NS is launched within DE.

In summary, nDE/njDE demand four new parameters: k, σ_{sh}, τ_3, and R. The complex dependencies between these parameters are analyzed in the next section.

4. Experiments and results

The original DE is known as a good global optimizer, insensitive to getting stuck into a local optima. Also, jDE has been proven as an excellent tool for solving the global optimization problems [2]. Therefore, the primary challenge of this study was to show that the behavior of these algorithms can even be improved by using the NS. The goals of our experimental work were three-fold:

- To find the optimal parameter settings for NC in nDE and njDE,
- To show that both hybridized algorithms, nDE and njDE, improve the results of their original counterparts,
- To show that these algorithms are also comparable with the other state-of-the-art algorithms, like L-Shade and MVMO.

The DE parameters were configured as follows: the amplification factor of the difference vector $F = 0.9$, and the crossover control parameter $CR = 0.5$. The mentioned values of parameters F and CR are used as starting values $F_i^{(0)}$ and $CR_i^{(0)}$ for $i = 1, \ldots, NP$ of the jDE algorithm. Let us notice that setting the parameter of the NS in both DE algorithms is a complex task and, therefore, was the subject of extensive experiments explained in the remainder of the paper. The population size was set to $NP = 100$ by all the DE algorithms in tests. It is a crucial size for population-based algorithms and has a great influence on their performance. Therefore, the most appropriate value of this parameter was found after extensive experiments that is valid for all the DE variants. 25 independent runs were launched per instance of each algorithm.

Here, the results of the following DE variants were compared: DE, jDE, nDE, and njDE. In order to make the comparison as fair as possible, the same number of fitness function evaluations (FEs) was considered as a termination condition in all algorithms, although they originally use the number of generations for this purpose.

The section is organized as follows. At first, the test suite of benchmark functions is described. Then, the measures needed for estimating the quality of algorithms are discussed. Next, the PC configuration on which the experiments were conducted is presented. Finally, the illustration of the results follows that is concluded with a discussion in which the behavior of hybridized DE algorithms is analyzed in detail.

4.1. Test suite

The CEC 2014 test suite (Table 1) consists of 30 benchmark functions that are divided into four classes:

- Unimodal functions (1–3),
- Simple multi-modal functions (4–16),
- Hybrid functions (17–22),
- Composition functions (23–30).

Unimodal functions have a single global optimum and no local optima. Unimodal functions in this suite are non-separable and rotated. Multi-modal functions are either separable or non-separable. In addition, they are also rotated and/or shifted.

Table 1
Summary of the CEC'14 test functions.

	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal functions	1	Rotated High Conditioned Elliptic Function	100
	2	Rotated Bent Cigar Function	200
	3	Rotated Discus Function	300
Simple Multimodal functions	4	Shifted and Rotated Rosenbrocks Function	400
	5	Shifted and Rotated Ackleys Function	500
	6	Shifted and Rotated Weierstrass Function	600
	7	Shifted and Rotated Griewanks Function	700
	8	Shifted Rastrigins Function	800
	9	Shifted and Rotated Rastrigins Function	900
	10	Shifted Schwefels Function	1000
	11	Shifted and Rotated Schwefels Function	1100
	12	Shifted and Rotated Katsuura Function	1200
	13	Shifted and Rotated HappyCat Function	1300
	14	Shifted and Rotated HGBat Function	1400
	15	Shifted and Rotated Expanded Griewanks plus Rosenbrocks Function	1500
	16	Shifted and Rotated Expanded Scaffers F6 Function	1600
Hybrid functions	17	Hybrid Function 1 ($N = 3$)	1700
	18	Hybrid Function 2 ($N = 3$)	1800
	19	Hybrid Function 3 ($N = 4$)	1900
	20	Hybrid Function 4 ($N = 4$)	2000
	21	Hybrid Function 5 ($N = 5$)	2100
	22	Hybrid Function 6 ($N = 5$)	2200
Composition functions	23	Composition Function 1 ($N = 5$)	2300
	24	Composition Function 2 ($N = 3$)	2400
	25	Composition Function 3 ($N = 3$)	2500
	26	Composition Function 4 ($N = 5$)	2600
	27	Composition Function 5 ($N = 5$)	2700
	28	Composition Function 6 ($N = 5$)	2800
	29	Composition Function 7 ($N = 3$)	2900
	30	Composition Function 8 ($N = 3$)	3000

To develop the hybrid functions, the variables are divided randomly into some subcomponents and then different basic functions are used for different subcomponents [20]. Composition functions consist of a sum of two or more basic functions. In this suite, hybrid functions are used as the basic functions to construct composition functions. The characteristics of these hybrid and composition functions depend on the characteristics of the basic functions.

The functions of dimensions $D = 10$, $D = 20$, and $D = 30$ were used in our experiments. The search range of the problem variables was limited to $x_{i,j} \in [-100, 100]$.

4.2. Measures for estimating the quality of algorithms

The results from the algorithms were evaluated according to five standard statistical measures: *Best*, *Worst*, *Mean*, *Median*, and *StDev* values. Friedman tests [13] were conducted in order to estimate the quality of the results obtained by various nature-inspired algorithms for global optimization statistically. The Friedman test is a two-way analysis of variances by ranks, where the statistic test is calculated and converted to ranks in the first step. Thus, the null hypothesis is stated assuming that medians between the ranks of all algorithms are equal. Here, a low value of rank means a better algorithm [6]. The second step is performed only if a null hypothesis of a Friedman test is rejected. In this step, the post-hoc tests are conducted using the calculated ranks.

According to Demšar [5], the Friedman test is a more safe and robust non-parametric test for comparisons of more algorithms over multiple classifiers (also datasets) that, together with the corresponding Nemenyi post-hoc test, enables a neat presentation of statistical results [26]. The main drawback of the Friedman test is that it makes whole multiple comparisons over datasets and, therefore, it is unable to establish proper comparisons between some of the algorithms considered [6]. Consequently, a Wilcoxon two paired non-parametric test is applied as a post-hoc test after determining the control method (i.e., the algorithm with the lowest rank) by using the Friedman test. On the other hand, the Nemenyi test is very conservative and it may not find any difference in most of the experimentations [14]. Therefore, the Nemenyi test is used for graphical presentation of the results, while the Wilcoxon test shows which of the algorithms in the test are more powerful. Both tests were conducted using a significance level of 0.05 in this study.

4.3. PC configuration

All runs were made on an IBM Lenovo using the following configurations:

Table 2
The best results of the njDE obtained by optimization of CEC 2014 functions of dimension $D = 20$.

Func.	Best	Worst	Mean	Median	StDev
1	80.95E−12	16.33E−06	752.47E−09	13.99E−09	3.25E−06
2	0	0	0	0	0
3	0	56.84E−15	2.27E−15	0	11.37E−15
4	161.47E−03	12.90E+00	1.40E+00	1.03E+00	2.42E+00
5	19.23E+00	20.07E+00	20.02E+00	20.05E+00	164.16E−03
6	0	1.96E+00	600.11E−03	200.88E−03	545.07E−03
7	0	7.40E−03	295.84E−06	0.00E+00	1.48E−03
8	0	0	0	0	0
9	4.28E+00	7.51E+00	5.66E+00	5.65E+00	816.08E−03
10	0	7.96E+00	2.62E+00	1.71E+00	2.93E+00
11	84.81E+00	439.20E+00	274.89E+00	280.39E+00	78.37E+00
12	60.96E−03	112.87E−03	89.20E−03	92.03E−03	12.96E−03
13	166.56E−03	283.28E−03	216.69E−03	205.85E−03	38.14E−03
14	165.63E−03	396.04E−03	270.98E−03	275.17E−03	44.48E−03
15	1.14E+00	2.08E+00	1.63E+00	1.61E+00	205.26E−03
16	1.79E+00	3.81E+00	3.28E+00	3.38E+00	412.83E−03
17	6.65E+00	393.53E+00	160.12E+00	152.01E+00	105.04E+00
18	2.46E+00	11.45E+00	5.44E+00	4.59E+00	2.42E+00
19	1.72E+00	3.51E+00	2.44E+00	2.35E+00	408.26E−03
20	3.00E+00	8.74E+00	5.77E+00	5.79E+00	1.51E+00
21	1.38E+00	267.19E+00	92.54E+00	87.30E+00	75.19E+00
22	27.21E+00	77.78E+00	39.88E+00	38.89E+00	10.89E+00
23	330.06E+00	330.06E+00	330.06E+00	330.06E+00	58.02E−15
24	209.28E+00	213.65E+00	209.97E+00	209.67E+00	947.52E−03
25	203.44E+00	205.31E+00	203.87E+00	203.75E+00	432.05E−03
26	100.16E+00	100.31E+00	100.24E+00	100.23E+00	34.62E−03
27	300.00E+00	400.76E+00	385.61E+00	400.34E+00	29.01E+00
28	550.51E+00	766.44E+00	686.73E+00	668.29E+00	61.11E+00
29	240.26E+00	356.60E+00	275.59E+00	249.38E+00	43.18E+00
30	490.42E+00	984.32E+00	725.08E+00	719.74E+00	153.43E+00

1. Processor - Intel Core i5-7400 3.00 GHz $\times$ 4
2. RAM - 8 GB
3. Operating system - Linux Mint 19 Cinnamon

All versions of the tested algorithms were implemented within the Eclipse Photon CDT Framework.

4.4. Results

The results of our experimental work are assembled into four subsections, as follows:

- The results of the best njDE run by optimizing the function benchmark suite of dimension $D = 20$,
- Searching for the best parameter setup,
- Comparing with the state-of-the-art algorithms,
- Discussion.

In the remainder of the paper, the results of the mentioned experiments are presented in detail.

4.4.1. The results of the best run

The results of the best run obtained by optimizing the CEC 2014 function benchmark suite of dimension $D = 20$ are illustrated in Table 2.

Although the functions of three different dimensions were taken into consideration, there only this dimension is selected due to the limitation of the paper length. Thus, the best results were obtained by the following parameter setting: $\sigma_{sh} = 10$, $k = 15$, $R = 1$, and $\tau_3 = 0.1$. However, this setting was found after extensive experimental work, as described in the next subsections.

As can be seen from Table 2, the results are observed according to the five standard statistical measures and each of the 30 functions in the benchmark.

4.4.2. Searching for the best parameter setup

The purpose of this experiment was to find the complex relations between the parameters introduced by the NS. There are four new parameters, as follows: the novelty area width σ_{sh}, neighborhood size k, replacement size R, and the learning rate τ_3. Two hybridized DE algorithms were included into this test, i.e., nDE, and njDE. In line with this, four experiments

Table 3
The statistical analysis of the results obtained by nDE/njDE according to parameter σ_{sh}.

D	σ_{sh}	nDE					njDE				
		Fri.	Nemenyi		Wilcoxon		Fri.	Nemenyi		Wilcoxon	
			CD	S.	p-value	S.		CD	S.	p-value	S.
10	5	3.52	[3.21,3.83]		0.00964	†	**3.27**	[2.96,3.58]	‡	∞	‡
	10	3.63	[3.32,3.94]		0.00181	†	3.64	[3.33,3.95]		0.05592	
	20	3.62	[3.31,3.93]		0.00122	†	3.5	[3.19,3.81]		0.01255	†
	30	3.86	[3.55,4.17]	†	0.00036	†	3.8	[3.49,4.11]		0.00107	†
	40	3.39	[3.08,3.70]		0.0197	†	3.58	[3.27,3.89]		0.00056	†
	50	**3.17**	[2.86,3.48]	‡	∞	‡	3.64	[3.33,3.95]		0.04363	†
20	5	3.65	[3.34,3.96]		0.06301		3.45	[3.14,3.76]		0.00391	†
	10	3.8	[3.49,4.11]		0.00368	†	**3.28**	[2.97,3.59]	‡	∞	‡
	20	**3.23**	[2.92,3.54]	‡	∞	‡	3.77	[3.46,4.08]		0.01101	†
	30	3.39	[3.08,3.70]		0.02938	†	3.53	[3.22,3.84]		0.00570	†
	40	3.58	[3.27,3.89]		0.00604	†	3.54	[3.23,3.85]		0.00621	†
	50	3.54	[3.23,3.85]		0.03836	†	3.73	[3.42,4.04]		0.00427	†
30	5	3.42	[3.11,3.73]		0.02872	†	3.6	[3.29,3.91]		0.01463	†
	10	3.37	[3.06,3.68]		0.20327		**3.24**	[2.93,3.55]	‡	∞	‡
	20	3.38	[3.07,3.69]		0.20897		3.30	[2.99,3.61]		0.07215	
	30	**3.29**	[2.98,3.60]	‡	∞	‡	3.90	[3.59,4.21]		0	†
	40	3.68	[3.37,3.99]		0.017	†	3.90	[3.59,4.21]		0	†
	50	3.68	[3.37,3.99]		0.017	†	3.90	[3.59,4.21]		0	†

were performed for each of the algorithms, in which three parameters are fixed, while the remaining one is varied in a predefined interval of feasible values. When the best value of each parameter is found and set as default for the next experiment, it can be assumed that a good approximation of the best parameter setting has been found. In this case, new knowledge about the behavior of the specific parameter was discovered in each experiment.

In the remainder of the paper, the influence of the specific NS parameter is analyzed in detail.

Influence of the novelty area width σ_{sh}. In the first test, the influence of the novelty area width σ_{sh} on the results of the optimization was discovered. Indeed, this parameter determines how far away the solution in set A must be from the observed solution in set B that the former can be included into the neighborhood of the latter and, thus, declared as the novelty. This parameter is very problem-specific and, therefore, finding its proper value is a subject of extensive tests.

In our study, the parameter was varied in the interval $\sigma_{sh} = \{5, 10, 20, 30, 40, 50\}$ for each of the three observed dimensions. For both proposed algorithms, the other three NS parameters were fixed as follows: $k = 10$, $R = 1$, and $\tau_3 = 0.1$. In summary, there was $6 \times 25 \times 3 \times 2 = 900$ total runs of hybridized DE algorithms necessary.

The results of the experiment are illustrated in Table 3 that is divided into two parts corresponding to each of the observed hybridized DE algorithms. Each part is divided according to different dimensions. The results in the table present ranks obtained after the Friedman test and two corresponding post-hoc tests, i.e., Nemenyi, and Wilcoxon. Let us notice that the control algorithms are denoted in the table with a double dagger sign ('‡'), while the statistical significance between the control and the observed algorithms with one dagger ('†').

From the table, it can be seen that the best results (denoted in bold case) of the nDE were obtained by $\sigma_{sh} = 50$, $\sigma_{sh} = 20$, and $\sigma_{sh} = 30$ by solving the functions of dimensions $D = 10$, $D = 20$, and $D = 30$, respectively. The situation is slightly different for the njDE, where the lower values of $\sigma_{sh} = 5$, $\sigma_{sh} = 10$, and $\sigma_{sh} = 10$ are necessary for obtaining the best results by solving the functions of the same dimensions.

Influence of the neighborhood size *k*. In the second experiment, the influence of the neighborhood size k was experienced. Also this parameter has a crucial impact on the results of the optimization. Obviously, its proper value was determined after huge experimental work. In line with this, the parameter was varied in the interval $k = \{5, 10, 15, 20, 30, 50\}$, two parameters were fixed as $R = 1$, and $\tau_3 = 0.1$, while the best values of parameter σ_{sh}, found in the last experiment, were applied to the observed hybrid DE algorithms. In summary, we also needed to conduct $6 \times 25 \times 3 \times 2 = 900$ the total independent runs.

The results of the experiments are aggregated in Table 4, where they are presented as ranks obtained after the Friedman statistical test as well as corresponding results of the Nemenyi and Wilcoxon post-hoc statistical tests obtained after solving the benchmark functions of three observed dimensions.

Interestingly, the results show that the best neighborhood sizes for nDE are $k = 10$, $k = 15$, and $k = 5$, when solving the benchmark functions of the dimensions $D = 10$, $D = 20$, and $D = 30$, respectively, and for njDE $k = 10$, $k = 15$, and $k = 15$, when solving the benchmark functions of the same dimensions. Let us mention that the best results are presented in bold in the table. It seems that both the hybrid DE algorithms do not allow large neighborhoods.

Influence of the replacement size *R*. The third experiment was dedicated for recognizing the influence of the parameter replacement size R on the results of the observed hybrid DE algorithms. This parameter determines the number of novelty solutions by replacing the original solutions. In this test, the values of the parameter were varied in the interval

Table 4
The statistical analysis of the results obtained by nDE/njDE according to parameter k.

D	k	nDE					njDE				
		Fri.	Nemenyi		Wilcoxon		Fri.	Nemenyi		Wilcoxon	
			CD	S.	p-value	S.		CD	S.	p-value	S.
10	5	3.28	[2.97,3.59]		0.18141		3.3	[2.99,3.61]		0.119	
	10	**3.27**	[2.96,3.58]	‡	∞	‡	**3.15**	[2.84,3.46]	‡	∞	‡
	15	3.58	[3.27,3.89]		0.05938		3.56	[3.25,3.87]		0.01578	†
	20	3.48	[3.17,3.79]		0.08851		3.8	[3.49,4.11]	†	4.00E−05	†
	30	3.32	[3.01,3.63]		0.30854		3.36	[3.05,3.67]		0.00272	†
	50	3.66	[3.35,3.97]		0.00139	†	3.53	[3.22,3.84]		0.00154	†
20	5	3.61	[3.30,3.92]		0.03005	†	3.79	[3.48,4.10]	†	0.00114	†
	10	3.48	[3.17,3.79]		0.0951		3.28	[2.97,3.59]		0.08851	
	15	**3.40**	[3.09,3.71]	‡	∞	‡	**3.18**	[2.87,3.49]	‡	∞	‡
	20	3.54	[3.23,3.85]		0.15866		3.58	[3.27,3.89]		5.00E−05	†
	30	3.47	[3.16,3.78]		0.17619		3.46	[3.15,3.77]		0.01287	†
	50	3.56	[3.25,3.87]		0.03362	†	3.52	[3.21,3.83]		0.02559	†
30	5	**3.42**	[3.11,3.73]	‡	∞	‡	3.39	[3.08,3.70]	†	0.03074	†
	10	3.51	[3.20,3.82]		0.16853		3.45	[3.14,3.76]		0.02222	†
	15	3.60	[3.29,3.91]		0.01463	†	**3.38**	[3.07,3.69]	‡	∞	‡
	20	3.54	[3.23,3.85]		0.12924		3.47	[3.16,3.78]		6.95E−03	†
	30	3.55	[3.24,3.86]		0.04746	†	3.6	[3.29,3.91]		0.00159	†
	50	3.62	[3.31,3.93]		0.10383		3.59	[3.28,3.90]		0.00798	†

Table 5
The statistical analysis of the results obtained by nDE/njDE according to parameter R.

D	R	nDE					njDE				
		Fri.	Nemenyi		Wilcoxon		Fri.	Nemenyi		Wilcoxon	
			CD	S.	p-value	S.		CD	S.	p-value	S.
10	1	3.42	[3.11,3.73]		0.03144	†	1.78	[1.47,2.09]		0.02619	†
	2	3.92	[3.61,4.23]	†	4.70E-04	†	2.11	[1.80,2.42]		1.50E−04	†
	5	3.73	[3.42,4.04]	†	0.00256	†	**1.67**	[1.36,1.98]	‡	∞	‡
	10	3.55	[3.24,3.86]		0.00226	†	2.23	[1.92,2.54]		0.00187	†
	20	3.45	[3.14,3.76]		0.08076		2.89	[2.58,3.20]	†	0	†
	50	**3.11**	[2.80,3.42]	‡	∞	‡	4.33	[4.02,4.64]	†	0	†
20	1	2.98	[2.67,3.29]		0.01255	†	**2.20**	[1.89,2.51]	‡	∞	‡
	2	3.22	[2.91,3.53]		2.81E-02	†	2.58	[2.27,2.89]		0.04006	†
	5	**2.84**	[2.53,3.15]	‡	∞	‡	2.59	[2.28,2.90]		0.0028	†
	10	3.09	[2.78,3.40]		0.09012		2.54	[2.23,2.85]		0.00062	†
	20	3.01	[2.70,3.32]		0.11123		3.05	[2.74,3.36]	†	0	†
	50	3.82	[3.51,4.13]	†	0	†	4.15	[3.84,4.46]	†	0	†
30	1	**3.25**	[2.94,3.56]	‡	∞	‡	2.79	[2.48,3.10]		0.04846	†
	2	3.25	[2.94,3.56]		n/a	n/a	2.81	[2.50,3.12]		0.10749	
	5	3.25	[2.94,3.56]		n/a	n/a	2.92	[2.61,3.23]		0.06944	
	10	3.25	[2.94,3.56]		n/a	n/a	**2.64**	[2.33,2.95]	‡	∞	‡
	20	3.25	[2.94,3.56]		n/a	n/a	3.08	[2.77,3.39]		0	†
	50	3.55	[3.24,3.86]	†	0.05592		3.88	[3.57,4.19]	†	0	†

$R = \{1, 2, 5, 10, 20, 50\}$, the best values of parameters σ_{sh} and k found in the previous test serve as default while the learning rate was fixed to $\tau_3 = 0.1$. Thus, the number of independent runs was retained at the same level, i.e., $6 \times 25 \times 3 \times 2 = 900$.

The results of the statistical analysis are accumulated in Table 5, where they are divided according to the corresponding hybrid DE algorithms and dimensions of the benchmark functions. However, the Friedman statistical tests, together with Nemenyi and Wilcoxon post-hoc statistical tests, are used for evaluation of the algorithm quality. For nDE, the values $R = 50$, $R = 5$, and $R = 1$ demonstrate the best replacement sizes by optimizing the benchmark functions of dimensions $D = 10$, $D = 20$, and $D = 30$, respectively, while $R = 5$, $R = 1$, and $R = 10$ are the best values for the njDE by solving the functions of the same dimensions. Interestingly, the results obtained by the nDE using the replacement size from $R = 1$ to $R = 20$ by solving the functions of dimension $D = 30$ demonstrate the same results. This means that the values of the Wilcoxon post-hoc statistical test are not applicable as designated with sign 'n/a' in the table. The reason for this phenomenon is dealt with in the discussion later in the paper.

Influence of the learning rate τ_3. The last experiment deals with the influence of the learning rate τ_3 on the results of the hybrid DE algorithms. The parameter controls the number of applications of the NS within the nDE/njDE algorithms, and also has a crucial impact on the results of the optimization. Its appropriate value was found after extensive experimental

Table 6
The statistical analysis of the results obtained by nDE/njDE according to parameter τ_3.

D	τ_3	nDE					njDE				
		Fri.	Nemenyi		Wilcoxon		Fri.	Nemenyi		Wilcoxon	
			CD	S.	p-value	S.		CD	S.	p-value	S.
10	0.0	3.34	[3.03,3.65]		0.01743	†	3.72	[3.41,4.03]	†	0	†
	0.1	**3.09**	[2.78,3.40]	‡	∞	‡	**2.59**	[2.28,2.90]	‡	∞	‡
	0.2	3.85	[3.54,4.16]	†	0.00030	†	3.04	[2.73,3.35]		5.00E−05	†
	0.3	3.56	[3.25,3.87]		0.01426	†	3.28	[2.97,3.59]		0	†
	0.4	3.64	[3.33,3.95]		0.00776	†	3.61	[3.30,3.92]	†	0	†
	0.5	3.16	[2.85,3.47]		0.04551	†	4.34	[4.03,4.65]	†	0	†
20	0.0	3.56	[3.25,3.87]		0.03288	†	3.86	[3.55,4.17]	†	5.00E−05	†
	0.1	3.42	[3.11,3.73]		0.02442	†	**3.01**	[2.70,3.32]	‡	∞	‡
	0.2	3.73	[3.42,4.04]		0.00064	†	3.48	[3.17,3.79]		0.02680	†
	0.3	3.45	[3.14,3.76]		0.10204		3.31	[3.00,3.62]		0.08226	
	0.4	3.62	[3.31,3.93]		0.07493		3.2	[2.89,3.51]		0.00427	†
	0.5	**3.28**	[2.97,3.59]	‡	∞	‡	3.35	[3.04,3.66]		0.02118	†
30	0.0	**3.26**	[2.95,3.57]	‡	∞	‡	3.42	[3.11,3.73]	†	3.00E−05	†
	0.1	3.50	[3.19,3.81]		0.04363	†	**2.63**	[2.32,2.94]	‡	∞	‡
	0.2	3.70	[3.39,4.01]		0.04947	†	2.91	[2.60,3.22]		0	†
	0.3	3.73	[3.42,4.04]		0.01539	†	3.22	[2.91,3.53]		0	†
	0.4	3.82	[3.51,4.13]		0.00011	†	3.85	[3.54,4.16]		0	†
	0.5	3.89	[3.58,4.20]	†	0.00866	†	4.25	[3.94,4.56]		0	†

work, where the best parameter settings, as found in the previous three experiments, were set as default for σ_{sh}, k, and R. In order to understand better the behavior of the algorithms based on the parameter, the learning rate was varied in the interval $\tau_3 = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. This means that there the total $5 \times 25 \times 3 \times 2 = 750$ independent runs of the hybrid DE algorithms needed to be conducted.

The results of the statistical analysis are presented in Table 6, where, besides the five mentioned instances, also the instance of algorithms by using $\tau_3 = 0.0$ is included. Actually, this instance presents the original DE/jDE, where the NS was switched off.

As can be seen in the table, the best values of the parameter are $\tau_3 = 0.1$, $\tau_3 = 0.5$, and $\tau_3 = 0.0$ for nDE by optimizing the functions of dimensions $D = 10$, $D = 20$, and $D = 30$, respectively. On the other hand, the value $\tau_3 = 0.1$ represents the best parameter setting for optimizing functions of all dimensions by the njDE. Let us notice that the results of the original DE improve the results obtained using the nDE by optimizing the functions of dimension $D = 30$. However, this is a consequence of the phenomenon discussed in the last paragraph.

4.4.3. Comparative analysis

In this subsection, two comparative analyses were conducted, as follows:

- Comparing the nDE/njDE algorithms with their original counterparts DE/jDE,
- Comparing the best results obtained by the nDE/njDE algorithms in the last study with the other state-of-the-art algorithms, like L-SHADE and MVMO.

Thus, the purpose of the first comparative study was to show that the hybridization with NS improves the results of the original DE/jDE. In line with this, the results of different instances of the nDE and njDE are compared with their original counterparts by solving the functions of the dimensions $D = 10$, $D = 20$, and $D = 30$. The obtained results are evaluated by ranks obtained after the Friedman non-parametric statistical tests and their corresponding Nemenyi post-hoc tests.

The results of comparing the nDE and njDE algorithms obtained by varying neighborhood size $N \in \{5, 10, 15, 20, 30, 50\}$ with the results of the original DE and jDE by solving benchmark functions of dimension $D = 30$ are presented in Fig. 2(a). From Fig. 2, it can be seen that nDE improves the result of its hybrid counterpart, but not also the results of the jDE and njDE, while the results of njDE by all the other instances are substantially better than the results of the original DE and njDE algorithms (Fig. 2(b)).

Figs. 2(c)–2(d) illustrates the results of the nDE and njDE instances obtained by varying the replacement size $R \in \{1, 2, 5, 10, 20, 50\}$ that are compared with the original DE algorithms by solving the functions of dimension $D = 10$. As can be seen from Fig. 2(c), the nDE instances with replacement size $R = 1$ and $R = 50$ outperform the results of the original DE, but these were worse than those obtained by the jDE.

The other instances of the nDE algorithm are not comparable with the original DE. On the other hand, the njDE instance with replacement size $R = 5$ outperforms significantly the results obtained by the original DE and jDE. In summary, the njDE instances of the lower replacement sizes $R \leq 10$ are all comparable with the results of the original DE and jDE, while the results are worse, when the replacement sizes are increased (i.e., $R \geq 20$).

The results of the nDE and njDE instances obtained by varying the learning rate from the set $\tau_3 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ compared with the original DE (i.e., nDE instance with $\tau_3 = 0.0$) and jDE are also very interesting as can be seen

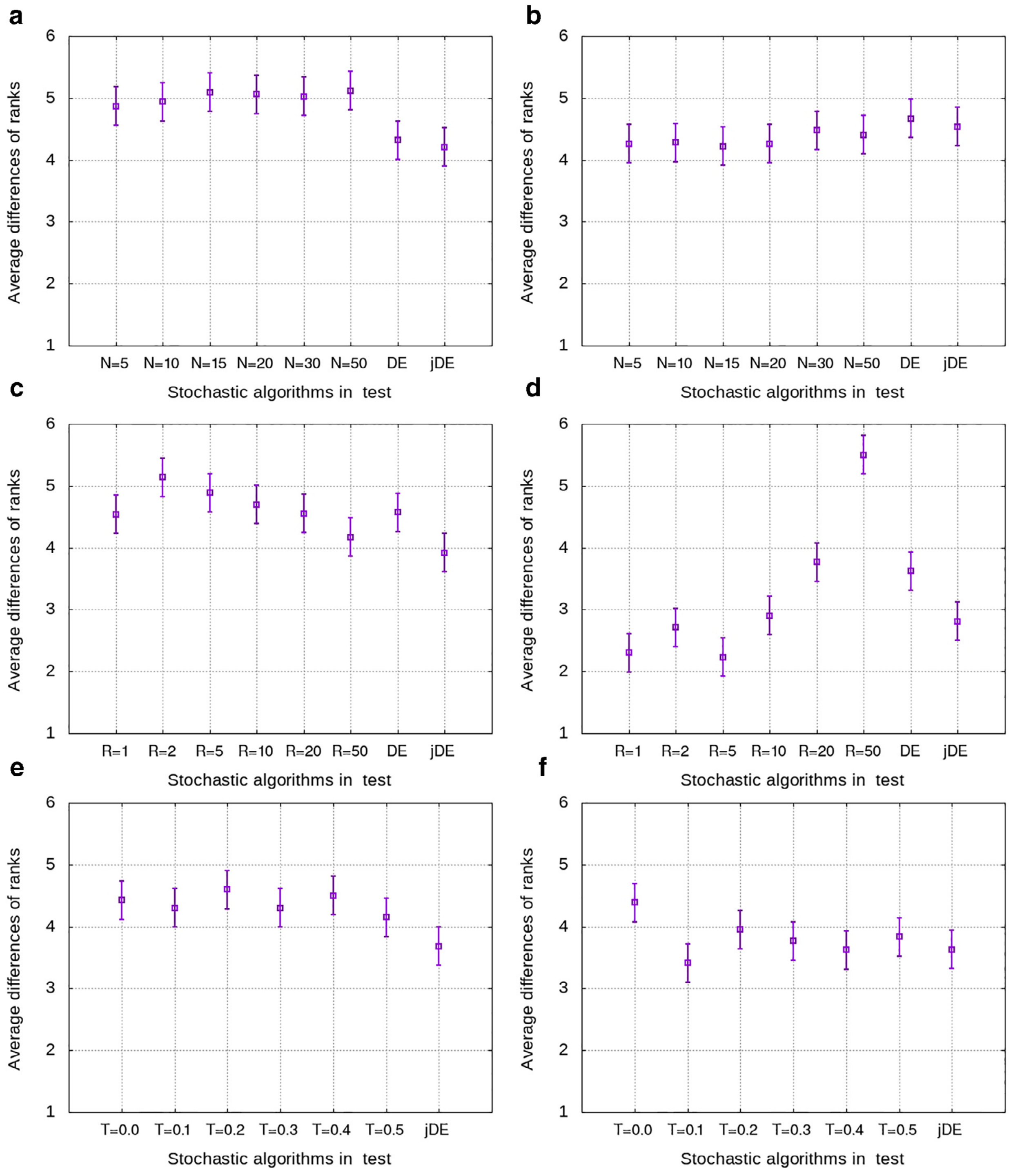

Fig. 2. The results of the Friedman non-parametric statistical tests. Top: nDE-variants by $D = 30$. Middle: nDE-variants by $D = 10$. Bottom: njDE-variants by $D = 20$.

in Figs. 2(e)–2(f). In Fig. 2(e), the best nDE instance with $\tau_3 = 0.5$ is the only one, whose results are not significantly worse when compare with the jDE. The situation is changed entirely, when the njDE algorithm is taken into consideration (Fig. 2(f)). Thus, the results of the njDE are comparable with those obtained by jDE by two instances, i.e., $\tau_3 = 0.1$ and $\tau_3 = 0.4$. In contrast, the original DE exposes the results that are significantly worse than those obtained by the three best algorithms in the comparative study.

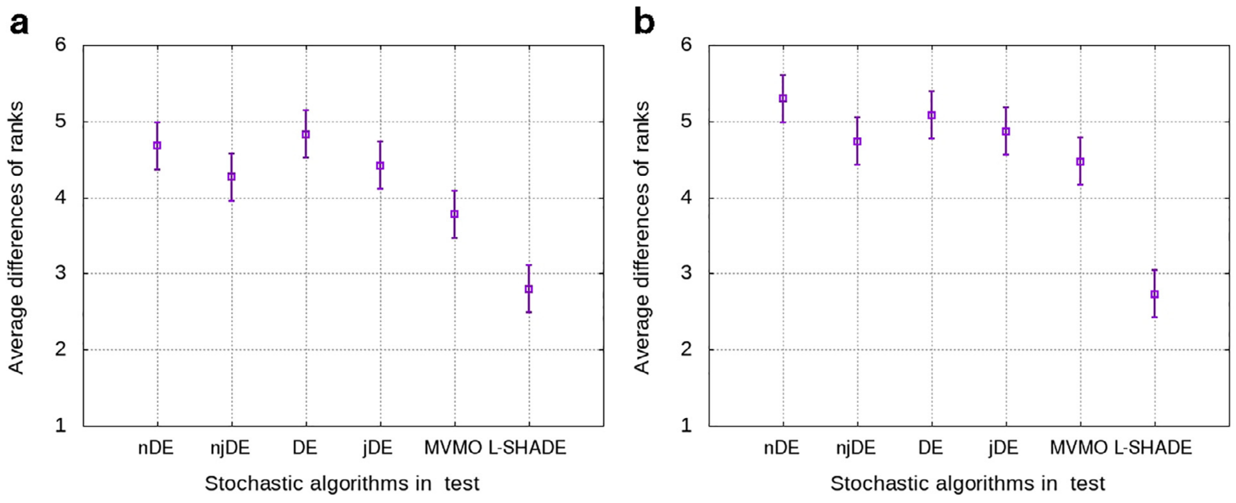

Fig. 3. Comparison with the state-of-the-art algorithms. Top: $D = 10$, Bottom: $D = 30$.

Table 7
The best parameter setting by nDE/njDE.

Dimension	nDE				njDE			
	σ_{sh}	N	R	τ_3	σ_{sh}	N	R	τ_3
10	50	10	50	0.1	5	10	5	0.1
20	20	5	15	0.1	10	15	1	0.1
30	30	5	10	0.1	10	15	10	0.1

The purpose of the second comparative study was to show that the results of the best njDE instance are also comparable with the results of the other state-of-the-art algorithms, like L-SHADE and MVMO. The results are also evaluated according to the results obtained after the Nemenyi post-hoc statistical tests and depicted in Fig. 3, where the former represents the results obtained by solving the functions of dimension $D = 10$, while the latter by solving the functions of dimension $D = 30$.

As can be seen from Fig. 3(a), the results of the L-SHADE are significantly better than those obtained by all the other algorithms in this comparative study. Moreover, the results of the MVMO are also significantly better than those obtained by the other algorithms except the njDE. The situation is slightly changed in Fig. 3(b), where the reported results of L-SHADE still remain the hard nut to crack for all the other algorithms in the test, while the difference between the results of the MVMO and the other algorithms become smaller.

4.4.4. Discussion

The results of our experimental work can be summarized in three findings, as follows:

- We tried to find the best parameter settings using the systematic search that are assembled in Table 7 with regard to different dimensions and different hybrid DE algorithms.
  Although we established that varying the parameter R within the nDE by solving the functions of dimension $D = 30$ does not have any influence on the results of the optimization, they definitely showed that this problem becomes too complex for the observed instances of the nDE.
- In general, the results of the original DE can always be improved by the results of the nDE. Additionally, the results achieved by the original jDE can be outperformed with the results obtained by the njDE. The exception presents instances of nDE when solving the benchmark functions of dimension $D = 30$.
- Comparison with the state-of-the-art algorithms showed that the results of the njDE are comparable with the results of the MVMO when solving the benchmark functions of dimension $D = 10$ as well as $D = 30$. Moreover, it is possible to find such parameter setting using an exhaustive search, with which the former is even capable of overcoming the results of the latter when solving the benchmark functions of dimension $D = 10$, as shown by Fister et al. in [11].

After performing an analysis of the experimental results, we were confronted with the following two issues:

- Why have the problems arisen with nDE instances when optimizing the benchmark functions of dimension $D = 30$?
- How does the NS affect the population diversity within the nDE/njDE algorithms during the optimization process and indirectly on the quality of the solutions?

In line with this, two additional tests were conducted, as follows:

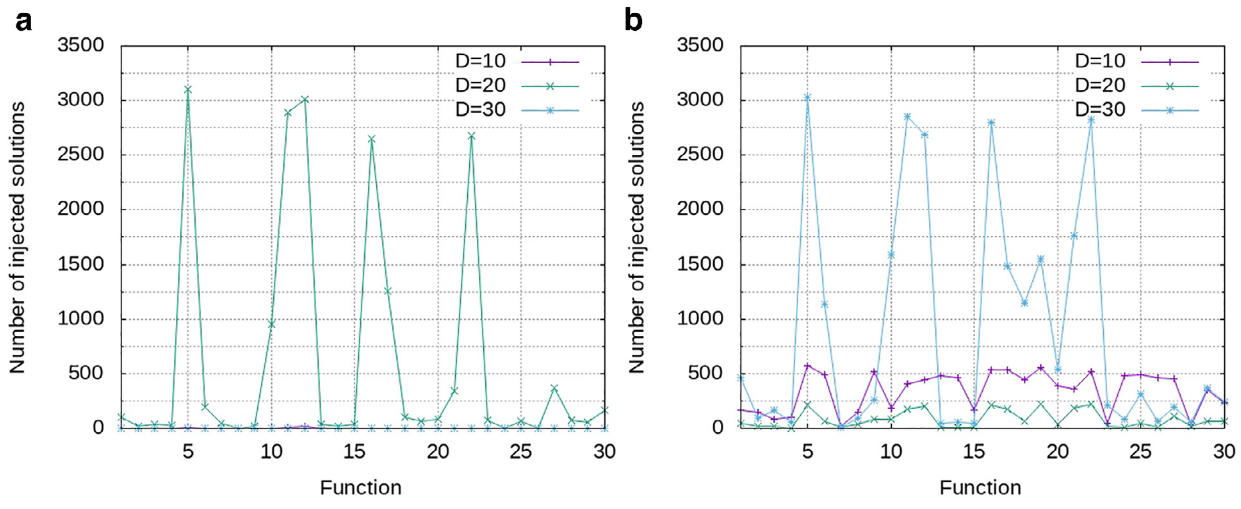

Fig. 4. The number of injected novelty solutions in one run. Top: nDE, Bottom: njDE.

- Counting the number of injected novelty solutions,
- Evaluating the influence of the population diversity.

The former answers the question of how the number of injected novelty solutions depends on the specific benchmark functions, while the latter the question of how the population diversity is changed during the optimization process. In the remainder of the paper, the mentioned tests are presented in detail.

Counting the number of injected novelty solutions. Interestingly, not all of the potential novelty solutions in set B satisfy the condition in Algorithm 1 (line 9) demanding that the NS metric must be higher than zero, in other words, $\rho(\mathbf{x}_i) > 0$. Satisfying this condition is a prerequisite for including the potential novelty solution $\mathbf{x}_i$ to the set of novelty solutions C, and is strongly dependent on the novelty area width σ_{sh}. The higher the area, the harder the novelty solution to be declared.

The goal of this test is to count the number of real novelty solutions from set C that can be injected into the current population. In line with this, all the injected novelty solutions which emerged in one typical run of the nDE/njDE are accumulated according to the specific benchmark functions, and illustrated in Fig. 4.

The figure is divided into two diagrams, which presented the results of the hybrid DE algorithms. Both diagrams depict three curves representing the number of injected novelty solutions according to the specific benchmark functions and their corresponding dimensions.

As can be seen from Fig. 4(a), there are a lot of injected novelty solutions only by those instances of nDE solving the benchmark functions of dimension $D = 20$. Rarely, the injections are emerged by solving the functions of the other dimensions. The situation is slightly different, when Fig. 4(b) is observed, where the behavior of the njDE algorithm is presented. From this figure, it can be seen that the injected novelty solutions have arisen by solving the benchmark functions of all dimensions. Interestingly, the majority of these solutions can be detected by solving functions of dimension $D = 30$. Additionally, the peaks are emerged by the same benchmark functions regardless of the observed algorithm.

Let us mention that the peaks actually denote the maximum number of injected solutions allowed in each NS call. For example, this number is calculated as $200 \times 15 = 3000$ for the nDE, and as $300 \times 10 = 3000$ for njDE, where the first value in the equation denotes the average call of the NS in one run, and the second is the parameter R. In contrast, there are no injected solutions for nDE by solving the benchmark functions of dimension $D = 30$ and only a few for $D = 10$.

In summary, two facts can be concluded from the tests: On the one hand, the number of injected novelty solutions depends on the observed function, while on the other, on the used algorithm. The first fact is connected directly with the parameter σ_{sh}, because each function describes its own fitness landscape This also means that distances on which a calculation of neighborhood are made, are specific. Consequently, the novelty area width that is appropriate for the function f_i is not appropriate for the function f_j and vice versa. Using the same value for parameter σ_{sh} for all the functions in the benchmark suite is, therefore, not optimal. However, when the figures are compared from the standpoint of different algorithms, it can be seen that the jDE is more appropriate for the hybridization with the NS than the DE due to generating more novelty solutions.

Finally, Fig. 4(a) also offers an answer to our first question, i.e., the problem emerged by optimizing the benchmark function of dimension $D = 30$ with the nDE. Because the number of injected novelty solutions is at most one, allowing it to replace more than one solution does not have any effect on the optimization.

Influence of the population diversity. The goal of this test was to discover the influence of the population diversity on the results of optimizing the CEC 2014 benchmark function suite. In line with this, two functions, precisely f_{22} and f_{27},

Table 8
The final results of the optimization.

D	Algorithm							
	nDE		njDE		DE		jDE	
	f_{22}	f_{27}	f_{22}	f_{27}	f_{22}	f_{27}	f_{22}	f_{27}
10	**0.0503**	2.5200	0.1052	**2.4701**	0.3687	300.0000	0.0759	2.9380
20	**24.7830**	400.2890	41.0955	400.4760	39.7045	**300.0000**	31.6990	400.4000
30	**45.2512**	355.6660	49.7878	354.4160	56.3622	**337.2350**	107.4430	400.9380

are taken into detailed analysis, because the characteristics of these functions are that they operate with a huge number of injected novelty solutions. Thus, we were focused on analyzing the behavior of both functions, varying their dimensions in interval $D \in \{10,20,30\}$.

As a measure of the population diversity, a moment of inertia I_c was used in our study as proposed by Morrison [23], and calculates the dispersion of each population member from their mass center. Mathematically, the mass center $\mathbf{c} = \{c_1, \ldots, c_D\}$ for $j = 1, \ldots, D$ is defined as follows:

$$c_j = \frac{1}{D}\sum_{i=1}^{Np} x_{i,j}. \tag{8}$$

The I_c is simple a sum of square distances from the mass center, in other words:

$$I_c = \sqrt{\sum_{i=1}^{Np}\sum_{j=1}^{D}(x_{i,j} - c_j)^2}. \tag{9}$$

The results of measuring the population diversity according to measure I_c in each generation of a typical evolutionary run are presented in Fig. 5, in which the optimization of the function f_{22} and f_{27} is conducted.

The figure is divided into six diagrams, where the population diversity is tracked according to both functions and their corresponding dimensions. In each diagram, a comparison of four algorithms is performed, as follows: nDE, njDE, DE, and jDE.

From the diagrams of functions f_{22} and f_{27} it can be seen that losing population diversity is the most evident for all algorithms optimizing the functions of dimension $D = 10$ (Figs. 5(a)–5(b)), where this measure decreases from the starting value towards zero. This phenomenon is the most expressive for the jDE algorithm, as can be seen in the graph of f_{22}. By optimizing the same function of higher dimensions (i.e., $D = 20,$ and $D = 30$), this is not so distinct, although also there the population diversity is lost the most by the jDE algorithm.

Interestingly, diagrams, illustrating a loss of the population diversity by optimizing function f_{27}, show that the mentioned phenomenon is the most observable for the original DE (Figs. 5(b), 5(d) and 5(f)), where the I_c measure falls to zero already after the some generations. On the other hand, the jDE is also capable of maintaining the population diversity in matured phases of the evolutionary search process.

The primary goal of this test was to show how the population diversity affects the quality of the results. In line with this, the results obtained in a randomly selected run of the particular evolutionary algorithm, are aggregated and presented in Table 8.

In the table, the best results are presented in bold case. As can be seen from this table, the best result obtained by optimizing the function f_{22} of dimension $D = 10$ were obtained by the nDE algorithm, while the njDE outperformed the results of the other algorithms in the test by optimizing the function f_{27} of the same dimension. Although the best population diversity by optimizing both benchmark functions of higher dimensions was maintained by the jDE, this algorithm did not achieve the best results.

In summary, higher population diversity does not ensure that the best results are achieved. Actually, the higher population diversity can prevent premature convergence, and directs the evolutionary search process to new undiscovered regions of the search space, where also exists huge potential for finding the best solution.

As shown from results of the experiments, the jDE is capable of maintaining the population diversity inherently. The NS adds an additional mechanism for maintaining this explicitly.

5. Conclusion

Nature-inspired algorithms have gained a new momentum, with huge development of ER and SR. This so-called nature-inspired robotics demands embodying these algorithms into hardware, where they act as autonomous artificial agents. New problems have arisen by moving them into the hardware, where the selection pressure is one of the topical issues. The higher selection pressure affects losing population diversity that is a prerequisite for the open-ended evolution and Alife. The NS presents one of the fundamental solutions for preserving the population diversity, because it introduces an additional measure for evaluating the individual's quality besides the fitness function.

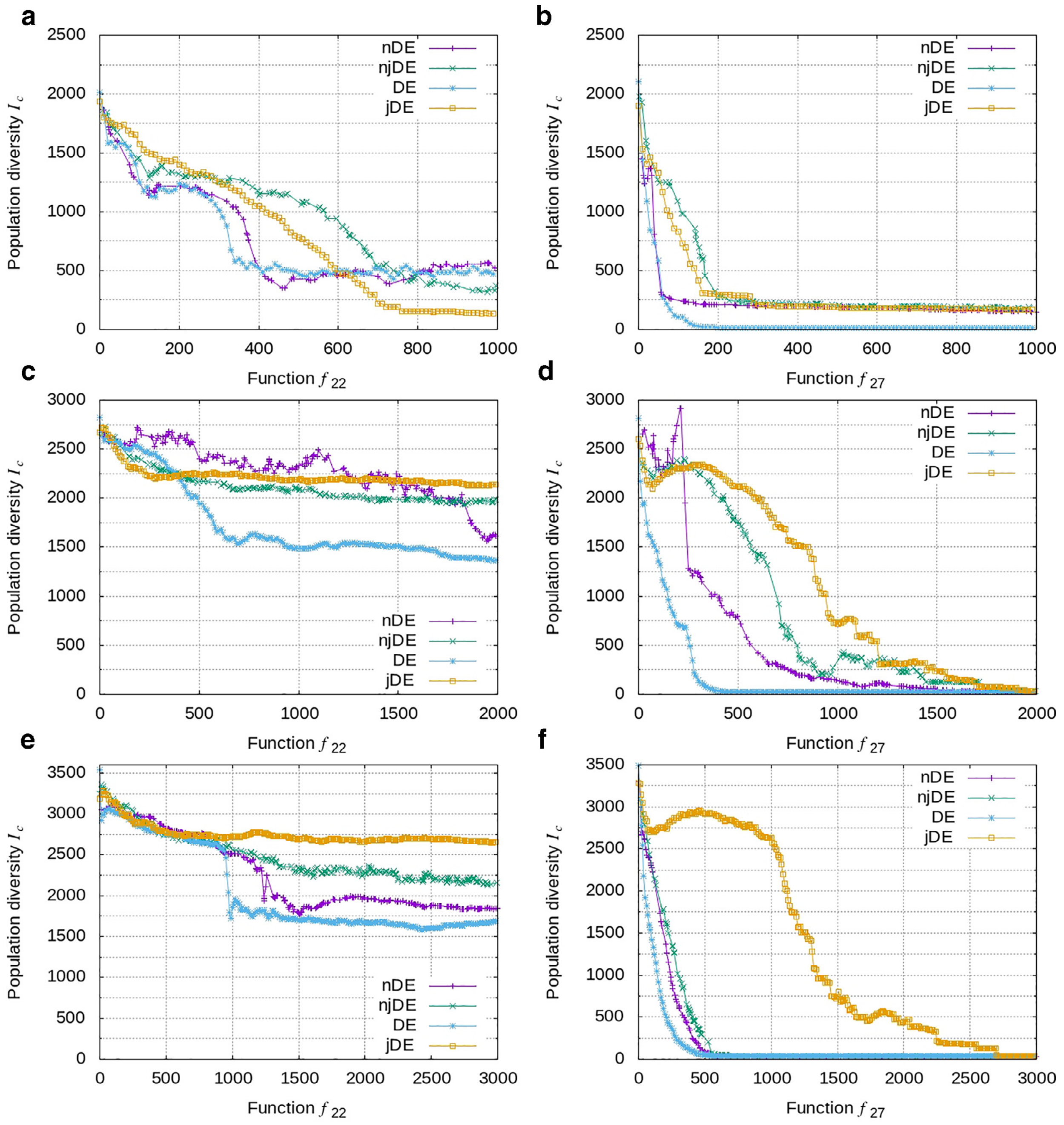

Fig. 5. Maintaining the population diversity by different stochastic nature-inspired population-based algorithms solving the benchmark functions of different dimensions. Top: $D = 10$, Middle: $D = 20$, Bottom: $D = 30$.

In our study, the original DE and the self-adaptive jDE were hybridized with the NS due to explicit maintaining of the population diversity and preventing the fitness function deception. As a result, two versions of hybrid DE algorithms were developed, i.e., nDE and njDE. The CEC 2014 benchmark function suite served as a testbed for testing the quality of the developed algorithms. Indeed, the functions of three different dimensions $D = 10$, $D = 20$, and $D = 30$ were taken into consideration.

During the experimental work, a huge number of experiments were conducted, in which searching for the best parameter settings of the hybrid DE algorithms was performed at first. These algorithms support four new parameters that have crucial impact on the results of the optimization. The best parameter settings were discovered using the systematic search, where it was found out that the nDE algorithm is insensitive on the replacement size parameter by solving the benchmark functions of dimension $D = 30$. Then, the comparative analysis, in which the results of the hybrid algorithms were compared with the other state-of-the-art algorithms, like L-Shade and MVMO.

In summary, the results showed that the nDE improves the results of its original DE counterpart, except for solving the benchmark functions of dimension $D = 30$. On the other hand, the njDE overcame the results of the original jDE by solving the benchmark functions of all dimensions. The comparative study showed that the L-Shade remains the best algorithm, while the njDE produced results comparable with the MVMO.

Finally, the insensitiveness of the nDE on the replacement size parameter R by solving the benchmark functions of dimension $D = 30$ was discovered. In line with this, two tests were performed: counting the number of injected novelty solutions, and evaluating the influence of the population diversity during the typical run. The results of the former test showed that the number of injected novelty solutions falls beyond the value of replacement size parameter by solving all the benchmark functions. Consequently, this parameter does not have any influence on the results of the optimization. On the other hand, the number of injected solutions depends on the observed function. Therefore, using the same value of novelty area width is not appropriate for all the benchmark functions. The latter test showed that maintaining the population diversity does not also lead automatically to the best solution, but it is a prerequisite for the open-ended evolution. In this sense, the njDE has a great potential for use in environments that demand an open-ended evolution.

As the future work, the self-adapting of the novelty area width parameter σ_{sh} might be realized at first. Thus, all benchmark functions could have the same chances to operate with the novelty solution and not only those, for which this value is selected properly by chance. Additionally, the impact of the NS on the multiagent systems based on DE could become a big challenge direction.

Acknowledgments

Iztok Fister acknowledges financial support from the Slovenian Research Agency (Grant no. P2-0041). Iztok Fister Jr. acknowledges financial support from the Slovenian Research Agency (Grant no. P2-0057). Matjaž Perc acknowledges financial support from the Slovenian Research Agency (Grant nos. J1-7009, J4-9302, J1-9112 and P5-0027). Andres Iglesias and Akemi Galvez acknowledge financial support from the projects TIN2017-89275-R (AEI/FEDER, UE) and PDE-GIR (H2020, MSCA program, ref. 778035). Eneko Osaba and Javier Del Ser would like to thank the Basque Government for its funding support through the EMAITEK program.

References

[1] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, Trans. Evol. Comp. 10 (6) (2006) 646–657.
[2] J. Brest, A. Zamuda, I. Fister, B. Boskovic, Some improvements of the self-adaptive JDE algorithm, in: Proceedings of the 2014 IEEE Symposium on Differential Evolution (SDE), IEEE, 2014, pp. 1–8.
[3] C. Darwin, On the Origin of Species, Harvard University Press, London, 1852.
[4] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.
[5] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[6] J. Derrac, S. Garca, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[7] S. Doncieux, J. Mouret, Behavioral diversity measures for evolutionary robotics, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.
[8] S. Doncieux, J.-B. Mouret, Beyond black-box optimization: a review of selective pressures for evolutionary robotics, Evolut. Intell. 7 (2) (2014) 71–93.
[9] A.E. Eiben, J.E. Smith, From evolutionary computation to the evolution of things, Nature 521 (7553) (2015) 476–482.
[10] I. Erlich, J.L. Rueda, S. Wildenhues, F. Shewarega, Solving the IEEE-CEC 2014 expensive optimization test problems by using single-particle MVMO, in: Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1084–1091.
[11] I. Fister, A. Iglesias, A. Galvez, J.D. Ser, E. Osaba, I. Fister, Using novelty search in differential evolution, in: Javier, Bajo (Eds.), Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection, Springer International Publishing, Cham, 2018, pp. 534–542.
[12] I. Fister, M. Mernik, B. Filipič, Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm, Comput. Optim. Appl. 54 (3) (2013) 741–770.
[13] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[14] S. Garca, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (12) (2008) 2677–2694.
[15] J. Gomes, P. Mariano, A.L. Christensen, Avoiding convergence in cooperative coevolution with novelty search, in: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2014, pp. 1149–1156.
[16] J. Gomes, P. Mariano, A.L. Christensen, Devising effective novelty search algorithms: a comprehensive empirical study, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO, ACM, New York, NY, USA, 2015, pp. 943–950.
[17] M. Kimura, The Neutral Theory of Molecular Evolution, Cambridge University Press, Cambridge, 1983.
[18] J. Lehman, K.O. Stanley, Exploiting open-endedness to solve problems through the search for novelty, in: Proceedings of the Eleventh International Conference on Artificial Life (Alife XI), MIT Press, Cambridge, MA, USA, 2008, pp. 329–336.
[19] J. Lehman, K.O. Stanley, Abandoning objectives: evolution through the search for novelty alone, Evol. Comput. 19 (2) (2011) 189–223.
[20] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, China and Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, 2013. Technical report, Singapore
[21] A. Liapis, G.N. Yannakakis, J. Togelius, Constrained novelty search: a study on game content generation, Evol. Comput. 23 (1) (2015) 101–129.
[22] M. Lynch, The evolution of genetic networks by non-adaptive processes, Nature Rev. Genet. 8 (2007) 803–813.
[23] R.W. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, Springer-Verlag, Berlin, 2004.
[24] E. Naredo, L. Trujillo, Searching for novel clustering programs, in: Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation, GECCO, ACM, New York, NY, USA, 2013, pp. 1093–1100.

[25] A.L. Nelson, Embodied artificial life at an impasse. Can evolutionary robotics methods be scaled? in: Proceedings of the 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS), IEEE, Orlando, FL, USA, 2014, pp. 25–34.
[26] P. Nemenyi, Distribution-Free Multiple Comparisons, Princetown, NY, 1963 Ph.d. thesis.
[27] R.K. Standish, Open-ended artificial evolution, Int. J. Comput. Intell. Appl. 2 (2003) 167–175.
[28] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[29] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1658–1665.
[30] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45 (3) (2013) 1–35. 35::33