Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Survey Paper

Review of Differential Evolution population size

Adam P. Piotrowski

Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, 01-452 Warsaw, Poland

ARTICLE INFO

Article history:
Received 2 June 2015
Received in revised form
29 January 2016
Accepted 31 May 2016
Available online 4 June 2016

Keywords:
Differential Evolution
Population size
Adaptive control parameters
Evolutionary Algorithms
Metaheuristics

ABSTRACT

Population size of Differential Evolution (DE) algorithms is often specified by user and remains fixed during run. During the first decade since the introduction of DE the opinion that its population size should be related to the problem dimensionality prevailed, later the approaches to DE population size setting diversified. In large number of recently introduced DE algorithms the population size is considered to be problem-independent and often fixed to 100 or 50 individuals, but alongside a number of DE variants with flexible population size have been proposed.

The present paper briefly reviews the opinions regarding DE population size setting and verifies the impact of the population size on the performance of DE algorithms. Ten DE algorithms with fixed population size, each with at least five different population size settings, and four DE algorithms with flexible population size are tested on CEC2005 benchmarks and CEC2011 real-world problems. It is found that the inappropriate choice of the population size may severely hamper the performance of each DE algorithm. Although the best choice of the population size depends on the specific algorithm, number of allowed function calls and problem to be solved, some rough guidelines may be sketched. When the maximum number of function calls is set to classical values, i.e. those specified for CEC2005 and CEC2011 competitions, for low-dimensional problems (with dimensionality below 30) the population size equal to 100 individuals is suggested; population sizes smaller than 50 are rarely advised. For higher-dimensional artificial problems the population size should often depend on the problem dimensionality d and be set to $3d$–$5d$. Unfortunately, setting proper population size for higher-dimensional real-world problems ($d > 40$) turns out too problem and algorithm-dependent to give any general guide; 200 individuals may be a first guess, but many DE approaches would need a much different choice, ranging from 50 to $10d$. However, quite clear relation between the population size and the convergence speed has been found, showing that the fewer function calls are available, the lower population sizes perform better.

Based on the extensive experimental results the use of adaptive population size is highly recommended, especially for higher-dimensional and real-world problems. However, which specific algorithms with population size adaptation perform better depends on the number of function calls allowed.

1. Introduction

Among three main control parameters of Differential Evolution (DE) [128] optimization algorithms the significance of scale factor (*F*) and crossover rate (*CR*) has already been deeply researched – see the review in [31,99]. In many recently introduced DE algorithms the values of *F* and *CR* do not need to be specified by the user, but are adapted or self-adapted during run [12,48,66,75,81,82,92,118,124,171]. The impact of the third control parameter, population size (*PS*), on the performance of DE algorithms has been rarely studied so far, and to motivate the choice of *PS* in many papers the reader is referred to [50], an interesting but old and very brief study. This may be surprising, as in case of other population-based Evolutionary Algorithms (EA) *PS* attracted large attention both in empirical and theoretical studies [2,22,39,40,58,76,84,88,93]. In the overwhelming majority of DE algorithms (as well as other EAs [41]) *PS* needs to be pre-specified and is kept fixed during run [31]. Researchers that propose novel DE methods suggest setting *PS* to very different values (differences exceed an order of magnitude – see [31,98,121]), but in a few studies [3,28,47,61,96,119,138,148] these choices are backed by the analysis of the impact of *PS* on the performance of the proposed algorithm. Although in recent years a number of DE algorithms with variable *PS* have been proposed [14–16,43,51,107,131,134,136,145,149,173,175], surprisingly such approaches have never been compared with each other (even more recently introduced DE algorithms with flexible *PS* are not compared with their older counterparts in the source papers) and their superiority over DE algorithms with fixed population size has not

E-mail address: adampp@igf.edu.pl

http://dx.doi.org/10.1016/j.swevo.2016.05.003
2210-6502/© 2016 Elsevier B.V. All rights reserved.

been verified. In fact, answers to the questions like 1. “how important is *PS* for the performance of DE algorithms”, 2. “how large *PS* should be”, 3. “should *PS* depend on the dimensionality of the problem to be solved”, 4. “should *PS* be fixed or modified during run”, 5. “what concepts the population size adaptation should follow” and 6. “how to relate *PS* with the number of allowed function calls” may today be only intuitive, due to the lack of detailed study on this topic. As the reader may expect, the above questions are addressed in the present paper.

To verify the impact of the population size on the performance of DE optimizers, and to find some rules how to choose DE population size, in this study 10 various DE algorithms are tested with 5–10 different but fixed population sizes on 2- to 50-dimensional CEC2005 benchmarks [130] and 1- to 216-dimensional CEC2011 real-world problems [30]. To understand and verify the efficiency of this few population size adaptation schemes that have so far been proposed, 7 variants of 4 DE algorithms with variable population size are tested on the same problems.

The scope of the study is limited to DE approaches developed for low- to moderately high-dimensional single-objective non-dynamic optimization problems. Due to the length of the manuscript, the main criterion for comparison of algorithms and their population sizes is the best objective function value found within the maximum number of function calls that has been defined for CEC2005 and CEC2011 problems in source publications [30,130]. However, in many practical applications convergence speed is an important factor; to get this into account the comparison of the results obtained after two much smaller numbers of function calls is also considered in this study.

2. Differential Evolution

The first population-based DE optimization method has been introduced in [128,129]. Although there are plenty of DE algorithms today [31], most of them follows basically the similar scheme. After initial random generation from the uniform distribution, in every generation g individuals $\mathbf{x}_{i,g} = \left\{x_{i,g}^1, \dots, x_{i,g}^d\right\}$, $i = 1, \dots, PS$ are evolved in order to find the vector $\mathbf{x}^*$ such that

$$f(\mathbf{x}^*) = \min_{\mathbf{x}\in\Omega\subseteq\mathbf{R}^d} f(\mathbf{x}) \tag{1}$$

when $f: \mathbf{R}^d \to \mathbf{R}$. The search is often restricted within the subset $\prod_{j=1}^{d}[L^j, U^j]$. The non-classical “*d*” notation, instead of “*D*”, is used in this study to facilitate noting visually the difference between “100” and “10*d*” that will frequently be used through the paper.

In each generation DE performs three steps called mutation, crossover and selection. Each individual, or parent ($\mathbf{x}_{i,g}$) creates first so-called donor vector ($\mathbf{v}_{i,g}$) by means of some mutation strategy. Plenty mutation schemes has been proposed so far [6,18,46,62,64,75,78,117,151,153], for example

DE/rand/1

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F\cdot(\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{2}$$

DE/best/2

$$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F\cdot(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) + F\cdot(\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g}) \tag{3}$$

DE/current-to-*p*best/1

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F\cdot(\mathbf{x}_{best,g}^{p} - \mathbf{x}_{i,g}) + F\cdot(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{4}$$

In above equations F is the control parameter called scaling factor, $r1$, $r2$, $r3$, and $r4$ are randomly selected integers from the range [1,*PS*], such that $r1 \neq r2 \neq r3 \neq r4 \neq i$, $\mathbf{x}_{\mathrm{best},g}$ is the best individual in the current population and $\mathbf{x}_{\mathrm{best},g}^{p}$ is a randomly selected individual from the top *p*% best individuals of the current population.

After mutation a crossover between donor ($\mathbf{v}_{i,g}$) and parent ($\mathbf{x}_{i,g}$) vector is performed to generate an offspring (or trial vector) ($\mathbf{u}_{i,g}$). Although there are few crossover schemes (see the detailed discussion in [5,63,86,148,159,167,172]), in the vast majority of DE algorithms a binomial crossover is used

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j} & \text{if} \quad rand_i^j(0,1) \le CR \quad or \quad j = j_{rand,i} \\ x_{i,g}^{j} & \text{otherwise} \end{cases} \tag{5}$$

where the value of the control parameter *CR* should be set within [0,1] interval, $rand_i^j(0,1)$ means a random number generated within [0,1] interval from the uniform distribution and $j_{rand,i}$ is a randomly selected integer from [1,*d*] range. At this point often some constraints or bounds handling approaches are applied.

After crossover the objective function is evaluated for $\mathbf{u}_{i,g}$, and according to the greedy selection only the better of the offspring-parent pair is passed to the next generation

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if} \quad f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases} \tag{6}$$

DE continues the search until the stopping criteria are met (that are frequently defined by setting the maximum number of function calls).

Many ideas how to improve DE algorithms have been proposed – they are not discussed here, for a review the reader is referred to [31,99]. However, some of DE modifications led to approaches that do not strictly follow the “classical” scheme given above. For example, distributed DE algorithms [1,7,32,109,110,155] divide the total DE population into sub-populations that may behave differently, and set the rules of communication, or migration of individuals, between them. Memetic DE methods combine DE paradigm with some local search procedures to speed up exploitation without loosing the global search capabilities [20,77,97,100,111]. DE has also been hybridized with many other heuristic algorithms [1,11,25,57,59,81,85,137,164,174], and was implemented as a part of multialgorithms [106,132,142–144]. As such approaches often do not follow DE scheme discussed above, the impact of the population size on their performance may depend on many specific features. Such “non-classical” DE concepts are not researched in this study.

3. Population size in Differential Evolution

Population size of DE must be set larger than the number of different randomly selected integers in the chosen mutation operation, otherwise difference vectors could not be constructed (consult Eqs. (2)–(4)). To understand the importance of the population size, some insight into the behavior of DE algorithms is needed. According to [49,155] DE does not require maintaining high population diversity during the whole search, although this opinion is not always acknowledged [165]. As in DE the step size depends mainly on the magnitude of difference vectors, and hence on the distance between individuals in the search space, DE diversity should decrease during run to allow ultimate convergence to some local optima. The dependence of DE on magnitude of difference vectors is frequently considered a main advantage of DE by practitioners, but on the other hand it does not allow proving the convergence of classical DE with probability 1 without adding some extra terms [71,72]. The scheme of DE behavior may be summarized as follows [49,95,155]. Initially individuals are randomly generated in the search space, hence distances between

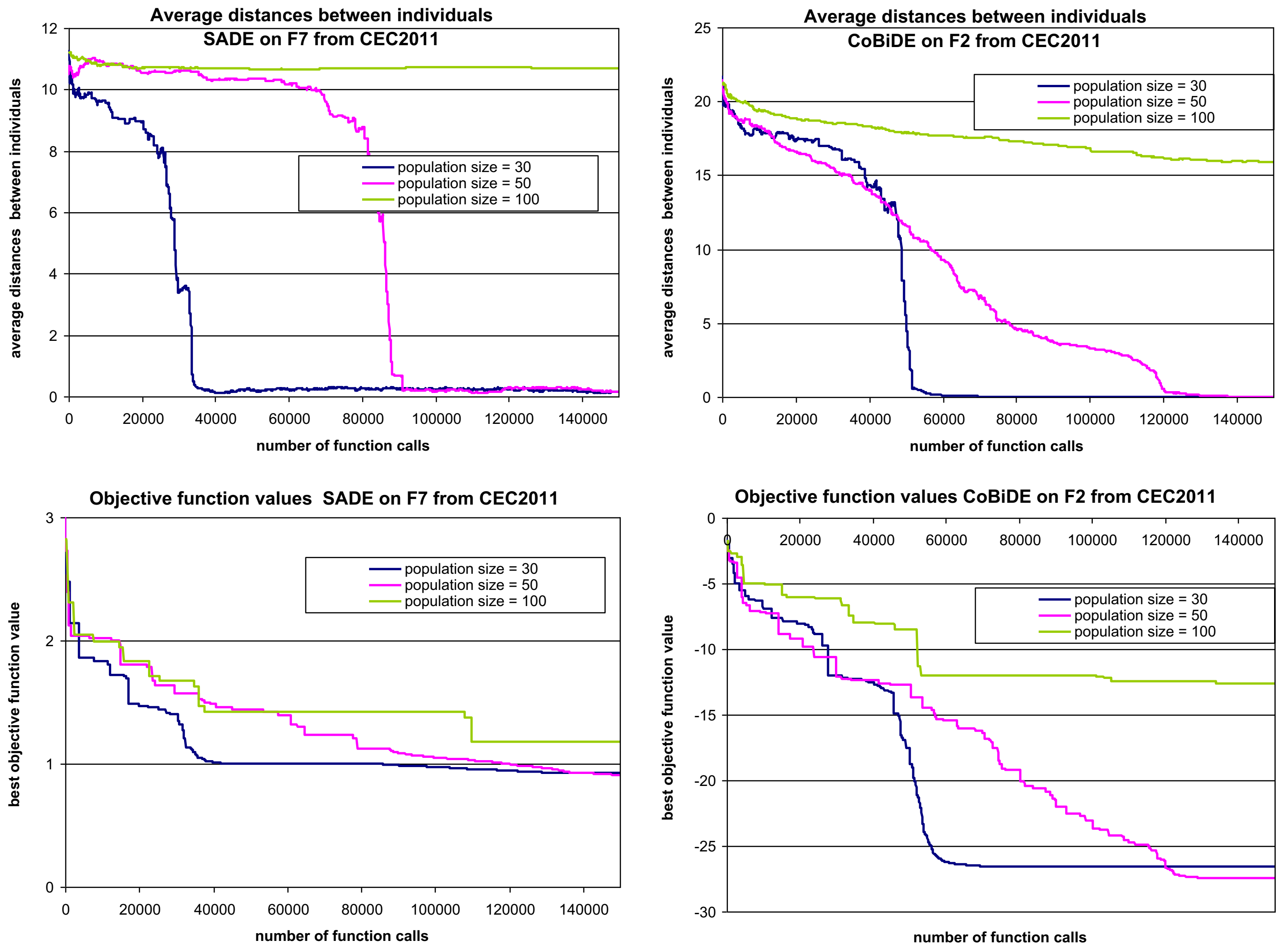

Fig. 1. Example illustration of convergence of population size of Differential Evolution algorithms. The picture shows the average distances between individuals during a single but representative runs of SADE and CobBiDE algorithms with various population sizes on two selected real-world problems from CEC2011 competition. F2 represent 30-dimensional Lennard-Jones potential problem and F7 – 20-dimensional radar system design problem. Note, that as each case shown in this picture is based on a single run only, the final objective function values cannot be compared with those based on averaged performance after many runs, discussed in further part of the study.

them are large. The probability of finding some individuals close to each other that could form difference vectors of low magnitude is low (although depend on the number of individuals and problem dimensionality), hence initially large exploratory steps prevail. Because DE follows greedy selection, when algorithm proceeds the individuals that survive often cluster in "better" parts of the search space. The distances between individuals within a cluster become small, but difference vectors of large magnitude are still easily obtained when the chosen individuals belong to different clusters, hence both large explorative and small exploitative steps may occur at this stage. As the time proceeds, the individuals are expected to concentrate around a few clusters and small exploitation steps become more frequent. Finally, if all individuals find themselves close to the same optimum, only the difference vectors of small magnitude are available and exploitation is performed. The clustering tendency of individuals in DE is illustrated in Fig. 1 for two different algorithms with three population size settings applied to solve two selected real-world problems from [30]. Although, as shown in [112], the scheme discussed above is not always fulfilled in practice, it allows understanding the specific impact of DE population size on the balance between exploration and exploitation (that differs from the one discussed for most EAs in [26]) and hence on the performance of DE. Using too small population size limits the number of available moves, what either, if individuals would not cluster, may lead to stagnation (the population stops proceeding toward the optimum, although the population diversity remains high [83], what may happen if different individuals become stuck close to different local optima), or premature convergence. On the contrary, very large population slow down clustering of individuals and frequently waste many function calls on almost random explorative moves (for an illustrative example, see Fig. 1). The public opinion is that small population sizes are beneficial for the separable or unimodal functions, while higher population sizes are needed for high-dimensional, non-separable problems [91,121]. One may also expect that the lower number of function calls is available, the smaller population size should be.

When introducing novel DE algorithms many authors specify how (and, more rarely, why) *PS* should be set, but rarely confirm their choice by any experiments. Unfortunately, in many papers one may find only information that "experiments were performed with population size set to X", without any discussion. In such a case the reader is left alone when applying particular DE algorithm to his specific problem (what is, unfortunately, not rare in EAs, see the discussion in [27]). In the literature the population size of DE algorithms is set mainly in four ways (for detailed discussion and references, see the next four sub-sections): 1. setting *PS* for each problem separately, without any motivation or according to the "expert knowledge", and keeping it fixed for the whole run; 2. relating *PS* to the dimensionality of the problem *d*, and keeping it fixed for the whole run; 3. setting the same fixed *PS* for all problems, irrespective of their dimensionality or properties; 4. allowing *PS* to vary during run, usually according to specified adaptation rules; often some bounds are set on *PS* variability.

3.1. Setting population size for every problem separately

Introducing DE [128], for each problem Storn and Price used different *PS* values, chosen by trial and error. Also in some much later studies different population sizes are sometimes chosen by hand for each considered problem [23,42,44,52,137,141,170]. However, this approach cannot satisfy needs of most users, as it either assumes that one has some extra knowledge on the properties of the problem to be solved, or a number of initial trials are performed to tune *PS* to the best value.

3.2. Problem dimensionality-dependent population size

As in DE the number of available moves depends on the population size, the obvious idea is to relate *PS* to the problem dimensionality *d*. In the first journal paper on DE [129] Storn and Price claimed that the choice of all three control parameters (*F*, *CR* and *PS*) is not difficult and suggested that *PS* should be set between 5*d* and 10*d*. Based on this hint *PS* equal 10*d* has been widely applied in many DE algorithms [4,5,29,80,87,108,123,154], and *PS* set to 5*d* has also sometimes been used [67,68,111]. In his next paper Price [116] suggested that *PS* could be as large as 20*d*. Although so large population sizes are often a burden for modern DE algorithms, according to [121] the best *PS* setting may vary even from 2*d* to 40*d*, depending on the problem. The first more detailed study on the impact of the population size on DE performance [50] narrowed the suggested bounds of *PS* to 3*d* and 8*d*, but this experimental research has been conducted on just a few simple benchmark problems and its results were not confirmed in a subsequent paper [90], according to which *PS* should depend both on the specific properties of the problem and on other control parameter values. From such early papers one could expect that rather larger population sizes are recommended, but in some DE algorithms developed later very small *PS* were used, for example equal to 1*d* [77,99,100]. Some authors [98,156] even suggested that *PS* should be set well below 1*d*.

In a few recently proposed DE algorithms the relation between *PS* and *d* is linear only within pre-set intervals. For example in [94] *PS* vary between 5*d* for over 10-dimensional problems, 10*d* for 5- to 10-dimensional problems and 20*d* when problem dimensionality is below 5. Also in [81,139] the higher dimensional problem, the lower the constant in the relation between *PS* and *d* is suggested (it is decreased from 5*d*–3*d* for low- to 2*d*–1*d* for high-dimensional problems). In [9] a bit surprising suggestion of setting *PS* to 1*d* if $d > 40$ or 4*d* otherwise may be found, according to which for $d=41$ the $PS=41$, but for $d=40$, *PS* would rise to 160.

3.3. Problem dimensionality-independent population size

As seen from the above discussion, the relation between *PS* and problem dimensionality in DE may be considered a mystery. Depending on the authors, *PS* may vary from below 1*d* [98] up to 40*d* [121]. As a result the simplest choices, like setting *PS* fixed and independent of *d*, what is frequent in case of e.g. Particle Swarm Optimization [38] and some other metaheuristics (to which we do not refer here after reading the work by Sorensen [127]), become tempting for DE. In a number of papers published after 2005 the population size of various DE algorithms has been simply set fixed to 100 [10–12,17,20,46,53,75,153] or 50 individuals [56,57,64,66,92,102,103,105,122,166], irrespective of the dimensionality of the problem. In some papers ideas of micro DE [21,101,120] – namely DE variants with extremely small *PS* – are tested; in the most extreme case [16] the *PS* may be set as low as 5. Accordingly, in many papers other fixed *PS* settings are suggested, like 10 [28], 20 [8,19,125], 30 [133,146], between 30 and 80 [138], 60 [7,152,157], 75 [82], 200 [158] or 250 [163]. Unfortunately, such choices seem rather intuitive and are rarely confirmed by competitive results with different population sizes. But even when the impact of fixed and problem-dimensionality independent *PS* on the performance of DE is tested, versatile results are reported. Some researchers [47,148] found that their algorithms perform best with $PS=30$, others [61,119] opted for few times larger population sizes. Recently an approach has been proposed [114] that uses small *PS* (set to 30), but chooses initial candidates from a large pool sampled in a specific way based on clustering before the actual algorithm commences. Obviously, specific DE algorithms may have different requirements, but as the number of DE approaches is large and in most cases their sensitivity to the population size has not been deeply researched, either some guide on the population size setting that could be roughly appropriate for many DE algorithms, or some adaptive population size approaches are highly needed.

3.4. Variable population size

Although the population size of DE algorithms is almost always set fixed, a few DE algorithms with variable population size have already been proposed, receiving probably less attention than they deserve.

The first two algorithms (with so-called absolute and relative encoding) that used self-adaptive *PS* have been proposed by Teo [134]. In both the initial *PS* is set to 10*d* and self-adapted after every generation at individual levels. Both proposed approaches use a specific three-parent's mutation strategy. However, Teo's idea was not widely explored in the subsequent studies, as he found that both proposed algorithms performed similarly to the basic DE on five rather easy test problems. The research was somehow repeated with the classical DE mutation strategies on 20 low-dimensional problems in [135], again showing comparable performance to the basic DE algorithm.

Other population size variability rule for DE has been explored in a number of papers, mostly co-authored by Janez Brest. Brest and Maucec [14] improved jDE algorithm with fixed *PS* [12] by introducing a concept of gradual decrease of population size. They followed the opinion that DE should rather explore the search space during initial stage, but when time proceeds more attention should be put on exploitation. The modified jDE algorithm, called dynNP-DE [14] is initialized with large population size (10*d*) and after each 25% of the allowed function calls *PS* is reduced by half, to 1.25*d* in the last quarter of the run. The choice of individuals to be deleted is based on the classical DE selection – the individuals (that are initially numbered) from the first and the second half of the population are compared in pairs and the poorer within the pair is discarded. The algorithm performed relatively well in further tests [150]. It was modified in [15] by introducing different mutation strategies for different parts of the population. In this novel approach, called jDElscop, the individuals that compete with each other during population reduction had to be evolved by the same mutation strategy. jDElscop outperformed three different EAs (including the classical DE) on 19 large-scale benchmark problems, and turned out best among over 20 EAs on non-classical tests aiming at maximization of CEC2005 functions [113]. A number of further modified versions of DE with *PS* reduction have been proposed in [16,168] that are in-depth discussed in a recent paper [169]. Similar concept of gradual decrease of *PS* has been implemented in [131]. However, in [131] the population is reduced after every generation linearly, starting from the assumed initial value (set to 20*d*) and ending with the smallest possible *PS* at which particular DE variant may run.

The population size adaptation mechanism based on the fitness diversity in the population has been introduced within FDSADE algorithm [136]. During FDSADE run at each generation the fitness diversity among individuals is calculated by some proposed measure. The population size is increased by duplicating selected best individuals if the fitness diversity decreases, or decreased by eliminating the worst individuals when the fitness diversity increases. According to [136] the *PS* of FDSADE should vary within [10,50] interval. In FDSADE the DE/rand/1 mutation strategy is used with *F* and *CR* being adapted according to the modified version of the mechanism introduced in [12] – the proposed modification is again based on the fitness diversity. In a very recent paper [107] the population size was suggested to depend on the complexity of the objective function, that would often also affect the fitness diversity in the population. Unfortunately, the proposed approach has been discussed for the basic DE variant only [107], and lead to very large *PS* values even for such simple functions like Rosenbrock's banana.

In few algorithms introduced in recent years specific *PS* adaptation rules were suggested – the population size is increased or decreased, depending on whether (or how many times) the algorithm improves the best solution within a few last generations. A number of such methods are briefly described below.

In DE algorithm proposed in [145] *PS* may vary within the predefined interval, set to [50,100]. If the best fitness value within the population is not improved during the pre-defined number of recent consecutive generations (*m*, set to 20), it is assumed that the population may be trapped in a local optimum and *PS* is increased by one individual. The novel individual is created by means of DE/best/1 strategy and, if *PS* is smaller than the maximum population size, enters the current population. If *PS* equals the maximum population size, the just-created individual competes with the worst individual in the current population according to the classical DE selection mechanism. On the other hand, if the population size is above its smallest allowed value and during the last *m* generations the best fitness value in the population is improved at least twice, to speed up the convergence *PS* is decreased by 1 by deleting the worst individual from the population. The algorithm described in [145] uses DE/rand/1 mutation strategy and rarely applied in practice exponential crossover (see [172]).

Wang and Zhao [149] in their SapsDE algorithm introduced a slightly more complicated population size adaptation mechanism, based on the concept contradictory to the one used in [145]. Wang and Zhao [149] proposed a single strategy of population size reduction and two different strategies to increase the population size, that may be used in various conditions. The idea was that *PS* should be modified quickly, depending on whether the best solution has been improved in the last generation or not. If the best fitness has been improved in the last generation, *PS* should be increased (not decreased, like in [145]) by 1, in order to facilitate exploration. The novel individual is created by means of DE/best/2 strategy. On the other hand, if the best fitness has not been improved during the last generation, but it has been improved during the last four generations, than *PS* is reduced by eliminating 1% (at least 1) of the worst individuals to speed up exploitation. Finally, if there was no improvement during the last 4 generations, or if the population size was not higher than its smallest value equal to 50 during the last 4 generations, the population is considered to be stuck in the local optima and *PS* is increased by 1% (at least 1 individual is added). In such a case the novel individuals are created by means of DE/rand/1 strategy. In SapsDE there is no upper limit of the population size. SapsDE uses *F* and *CR* adaptation approaches and the archive proposed in JADE [171], and applies (during regular part, not when *PS* is to be increased) DE/rand-to-best/1 and DE/current-to-*p*best/1 mutation strategies. The probability of applying particular mutation strategy at particular generation changes with the number of function calls exploited, such that at the beginning of the search DE/rand-to-best/1, and during later part of the run DE/current-to-*p*best/1 is more frequently used.

Another strategy of *PS* adaptation in DE, called ATPS-DE, was proposed in [175]. In ATPS-DE the so called status monitor counts the number of recent consecutive generations with or without improvement of the best solution. The upper and lower bounds of the population size are set (suggested to be [50,200]), but as they may be exceeded in some cases, the number of recent generations with the population size out of bounds is also counted by the status monitor. Based on the state of the status monitor variables, the strategies of decreasing or increasing population size may be implemented. The population size is decreased if either the improvement was noted during each of last 4 generations and the population size is not lower than its lowest allowed value, or the population size was higher than its highest allowed value during the last 4 generations. The population is increased if there was no improvement in the fitness of the best solution during last 4 generations and the population size is not higher than its highest allowed value, or the population size was lower than its lowest allowed value during last 4 generations. If the population size is to be decreased, the 1% of individuals are to be removed (however, due to the way the individuals to be deleted are selected, there is some low probability that in fact none individual will be removed from the population). The special ranking mechanism is performed, and the poorer rank is assigned to particular individual, the higher probability that it will be deleted. The best individuals (one or more, depending on the specific circumstances) are always kept in the population. If the population is to be increased, the novel individuals are created by applying a rather sophisticated mutation and crossover strategy to 1% of best individuals from the current population. Contrary to other papers that introduced novel population size adaptation schemes, in [175] the ATPS-DE strategy has been adopted to three different DE algorithms, namely jDE [12], CoDE [146] and JADE [171] – the last one (called ATPS-JADE) achieved the best performance.

The very similar to ATPS-DE population size adaptation scheme has been proposed within DE algorithm called SAPA [173]. The differences of both *PS* adaptation schemes are marginal and SAPA is again based on JADE algorithm.

One may find some clear similarities and differences between SapsDE and ATPS-DE (or SAPA). The similarities include: the use of a kind of status monitor variables; the modification of the population size if improvement is, or is not achieved within the last 4 generations; changing the population size by 1%. The main differences are: if there is no improvement for some time, the *PS* is decreased in SapsDE, but increased in ATPS-DE (and the algorithm proposed in [145]); in SapsDE there is no upper limit of *PS*; SapsDE applies two different *PS* increasing strategies, each in different conditions; the increasing and decreasing *PS* strategies used in ATPS-DE are much more complicated. Interestingly, SapsDE, ATPS-DE and SAPA were developed in the same institution within 2 years, but by different authors, and were not compared to each other. This show how diverse opinions on adaptation of population size may co-exist.

Another DE algorithm with variable population size has been introduced in [51] for geophysical applications. In this approach the population size is fixed during a number of initial generations, than it may be modified after every 3rd generation by putting into a kind of an archive some among individuals that were constantly ranked worst during last 3 generations, or by removing from the archive back to the main population a number of randomly selected individuals that did not participate in the evolution for some time. Unfortunately, the variable *PS* scheme proposed in [51] is applied to the very specific cooperative coevolution DE approach [147] and tested on just a single geophysical problem.

As various DE population size adaptation schemes frequently share some conceptual similarities, they may be classified into 4 groups. The first is based on the self-adaptation of *PS* at individual levels and include two algorithms [134,135]. The second group, which include algorithms developed in [14–16,131,168,169], is based on the heuristic idea of starting from the large population that facilitates initial exploration, and decreasing it gradually during run to intensify exploitation before termination. The third group refers to the concept exploited in just two papers [107,136], in which the adaptation of the population size depends on the fitness diversity of the algorithm, or the complexity of the problem that should also be linked with expected fitness diversity. The fourth group includes a number of specific population size adaptation strategies [145,149,173,175] that are based on the idea of changing the population size if one, more, or no improvements of the best solution are recorded during a number of recent generations. To some extent the approach proposed in [51] may also be classified into this group.

Finally, it must be mentioned that in some studies a much different kinds of DE with "non-fixed" population size have been proposed. For example Sarker et al. [124] introduced an algorithm in which the population size is switched between 75 and 100 individuals periodically, depending on the performances that variants with both population sizes are achieving. In DE algorithm proposed in [109,110] in specific circumstances some unneeded individuals may be "frozen", decreasing the population size – in further generations such individuals do not participate in the evolution, but may be used within difference vectors, like "archived" individuals in JADE. Weber et al. [155] proposed a distributed DE approach in which the size of some sub-populations is progressively reduced to enhance exploitation, but the size of other sub-populations is fixed in order to not hamper exploration. In [91] DE algorithm with three sub-populations of different sizes (2*d*, 4*d* and 8*d*) that compete for allowed number of function calls is proposed. Slightly similar approach has been introduced in [36] for dynamic optimization. Gonoguntla et al. [65] proposed to initially generate a large number of initial solutions and than in each generation perform DE operations only on some of them, keeping the rest unchanged. Wu et al. [163] introduced DE variant in which population is divided into four sub-populations, three of which use different mutation strategies and the 4th is allocated to the currently best performing sub-population – making the size of sub-populations that use particular mutation strategies variable, although the whole population size is fixed. However, as such specific algorithms do not introduce the novel population size adaptation methods that could alleviate the problem of the population size selection in versatile DE approaches, they are not exploited further in this study.

4. Comparison settings

To address the main questions of the present study a number of tests are performed with different DE algorithms and various population sizes. This section is devoted to brief discussion and motivation of the choice of 1. the problems used for comparisons, 2. DE algorithms with fixed population size, 3. the population sizes, and 4. DE algorithms with variable population size.

4.1. Problems used for comparison

Test problems used in this study include:

1. CEC2005 collection of 2-, 10-, 30- and 50-dimensional artificial benchmark problems [130];
2. CEC2011 collection of 22 non-scalable 1- to 216-dimensional real-world problems [30].

The best way to avoid the subjectivity when choosing problems on which tests are to be performed is to follow already existing and widely used collections [27]. As scalable problems are needed to find relations between the population size and the problem dimensionality, 25 scalable CEC2005 [130] artificially-constructed benchmarks are used in this study. However, the performance achieved on artificial problems may not necessarily be representative for evaluating optimization algorithms [113], hence CEC2011 [30] set of 22 real-world problems is also used. The results obtained for real-world problems are important to verify how general the conclusions based on artificial benchmarks are. The maximum number of function calls is set to 10,000*d* for CEC2005 problems and 150,000 for CEC2011, as specified in [30,130]. Both CEC2005 and CEC2011 include only minimization problems. Each tested DE variant is run 30 times on every problem.

In this study the performance of each DE variant is based mainly on the best (lowest) objective function value sampled within the maximum number of function calls specified in [30,130] (all results discussed in Sections 5.1–5.3 are based on such criteria). However, in a few recent papers [34,37,45] concerns regarding the use of maximum number of function evaluations as a stopping criteria were raised, and in some recent competitions (CEC2013, CEC2014) the performance of algorithms after various, sometimes a very few function calls is discussed. Also in many practical applications only low number of function calls may be allowed. Hence, although the usefulness of EAs to solve problems when number of function calls is low may be disputable (under such conditions EAs are inferior to mathematical modeling methods, consult [115]) in the Section 5.4 of the present study also the relation between the population size and convergence speed is considered.

For the purpose of this study from now on "DE variant" is understood as the particular DE algorithm with the specified rule of the population size setting (following this definition, when one DE algorithm is used with 7 *PS* settings, there are 7 DE variants). To avoid situation when some variants may be ranked better than the others due to marginally better performance achieved on the simplest problems, in this study it is assumed that ranks of DE variants are equal if the difference between their averaged performances is below 10^{-4}.

Following [33,55] the statistical significance of the results obtained is evaluated by means of Holm's correction [70] of Bonferroni-Dunn procedure [35] at 0.05 significance level. At first, DE variants are ranked for each problem, then their average rank over the whole set of problems is computed. DE variant with the lowest average rank for particular set of problems is chosen as the control method (for each comparison the control method may be different) and the statistical significances of the differences between each considered DE variant and the control method are tested. In some studies [54,140] testing the statistical significance of all-by-all comparisons of algorithms is suggested. Unfortunately, this approach cannot be followed in this study, as such statistical procedures are not recommended when very large number of variants are applied (consult [140] and program codes available therein), what is the case of the present research.

4.2. Differential Evolution algorithms with fixed population size

The following DE algorithms with fixed population size are used in this study:

1. CDE [17]; *PS* suggested in the source paper = 100;
2. MDE_pBX [75]; *PS* suggested in the source paper = 100;
3. EPSDE [92]; *PS* suggested in the source paper = 50;
4. SspDE [104]; *PS* suggested in the source paper = 100;
5. SADE [118]; in the source paper *PS* is suggested to be set by the user after initial trials; experiments in [118] are performed with *PS* = 50;
6. RB-SADE [61]; *PS* suggested in the source paper = 50 (when RB is merged with SADE);
7. AdapSS-JADE [60]; the algorithm with normalized average reward method is used, as the best among 4 ones proposed in [60]; *PS* suggested in the source paper = 100;
8. Rcr-JADE [63]; *PS* suggested in the source paper = 100;
9. JADE-EIG [67]; *PS* suggested in the source paper = $5d$;
10. CoBiDE [152]; *PS* suggested in the source paper = 60.

In the above list the population sizes suggested by the authors of each approach are given. Used in this study control parameters other than *PS* are the same as suggested in the source papers.

In this study it is assumed that the reader may be interested in the impact of *PS* on the performance of modern DE optimizers, hence only algorithms proposed during 2009–2015 period are tested. Almost all chosen DE algorithms with fixed population size (CDE is the exception) use adaptive or self-adaptive *F* and *CR* control parameters, what is a standard approach today. All 10 methods may be called classical DE approaches, as they do not use additional local searchers, are not hybridized with other optimizers (although CoBiDE uses the covariance matrix, it is not a DE and CMA-ES [69] hybrid) and do not follow the concepts of distributed, compact or cooperative coevolution computing. Non-classical DE algorithms are by purpose not applied in this study, as the impact of the population size on their performance may depend on too many factors not related to DE framework.

Most DE algorithms listed above are very different from each other, but a few have noticeable similarities. This choice has been made to address the following question: if some DE algorithms share many important features, but also have some clear differences, would the impact of *PS* on their performance be similar? To test that, three different extensions of JADE algorithm [171] are used (AdapSS-JADE, Rcr-JADE and JADE-EIG), as JADE-based approaches become abundant in recent years. However, to not boost the number of JADE-based methods, the original JADE is not tested here, and instead of RB-JADE, suggested as the best RB-based approach in [61], the RB-SADE (also introduced and discussed in [61]) is used. This allows verifying the impact of *PS* on two slightly different SADE algorithms. Finally, two methods (CoBiDE and JADE-EIG) that independently introduce the concept of rotating the coordinate system in order to improve the efficiency of the crossover operation in DE are tested. Hence, among tested algorithms there are three groups of approaches that share some important features: 1. AdapSS-JADE, Rcr-JADE and JADE-EIG; 2. SADE and RB-SADE; 3. CoBiDE and JADE-EIG.

The inventors of 5 among 10 used DE algorithms suggested *PS* to be fixed to 100, independently of the problem. Only in case of JADE-EIG the suggest *PS* has been related to the problem dimensionality. In contradiction to the earlier studies, in which DE population size usually depended on dimensionality of the problem (see Section 3.2), this seems to be a clear trend in recent years (see Section 3.3). The present study is hoped to verify if this choice is backed by something else than the simplicity.

4.3. Considered population sizes

As discussed in the Section 3, initially the population size of DE algorithms was often related to the problem dimensionality. Skipping a few extremes, this relation usually varied between *PS* = $10d$ and $1d$. However, in DE algorithms published during recent years the population size is frequently fixed to 100 or 50 individuals. Hence, in this study DE algorithms with fixed population size are tested on all problems with the following *PS* values:

1. *PS* = $1d$;
2. *PS* = $2d$;
3. *PS* = $3d$;
4. *PS* = $5d$;
5. *PS* = $10d$;
6. *PS* = 50;
7. *PS* = 100.

Note that in case of 10-dimensional CEC2005 problems *PS* = 50 and 100 are the same as *PS* = $5d$ and $10d$, respectively. Similarly, for 50-dimensional CEC2005 problems *PS* = 50 and 100 are the same as *PS* = $1d$ and $2d$, respectively. In case of 2-dimensional CEC2005 problems *PS* = $1d$ and $2d$ cannot be used, as such population sizes are too small – some mutation strategies used for example by SADE require 5 different individuals in the population. Hence, in these three cases only 5 different *PS* settings are tested.

In case of CEC2005 problems each comparison is based on functions of the same dimensionality (see Section 4.1). However, among non-scalable CEC2011 problems there are some very low- and some high-dimensional ones (the dimensionality vary between 1 and 216). To fairly test how DE algorithms with different *PS* settings perform in such cases, three additional problem-dimensionality independent population sizes are considered for CEC2011 problems, namely:

8. *PS* = 20;
9. *PS* = 30;
10. *PS* = 200.

This way on CEC2011 problems algorithms are tested with 5 dimensionality-independent population sizes, and 5 *PS* settings related to the problem dimensionality. As *PS* cannot be neither very small (this would result in too few possible moves) nor too huge (this would prevent convergence), in this study when DE algorithms with fixed but dimensionality-dependent population size are applied to real-world problems the *PS* can neither be smaller than 10 nor exceed 500. Hence, for CEC2011 problems in case of 10 DE algorithms with fixed *PS* the population size settings of, for example *PS* = $3d$, are understood as follows: if $3d < 10$ then *PS* = 10, else if $3d > 500$ then *PS* = 500, else *PS* = $3d$. Note that in case of DE variants with flexible population sizes the above bounds are not applied.

4.4. Differential Evolution algorithms with flexible population size

The following seven variants of four DE algorithms with flexible population size have been chosen in this study:

1. jDElscop [15]; initial *PS* = $10d$ is reduced progressively during run to $1.25d$; in this study the population size reduction is not performed if the current population size is below 9.
2. FDSADE [136]; *PS* may vary during run within [10,50].
3. FDSADE-A; *PS* may vary during run within [50,100].
4. FDSADE-B; *PS* may vary during run within [$1d$,$10d$]; in case of 2-dimensional CEC2005 problems *PS* cannot be smaller than 6; in case of CEC2011 problems additional bounds on *PS* are set at [10,500].

5. ATPS-JADE [175]; upper and lower bounds of *PS* are set to 50 and 200, but note that according to [175] they may be exceeded by the algorithm in specific circumstances for a few generations.
6. SapsDE [149]; lower bound of *PS* is set to 50; note that according to [149] it may be exceeded by the algorithm in some circumstances; no upper bound;
7. SapsDE-A; lower bound of *PS* is set to 50; upper bound is introduced and set to 300; note that both may be exceeded by the algorithm in some specific circumstances (analogically to the lower bound in the original SapsDE).

jDElscop is by definition initialized with $PS=10d$ [15]. Initial population size of other six variants is in this study set to $5d$, but within the *PS* bounds. jDElscop and SapsDE lacks upper bound of the population size, but in case of CEC2011 problems in this paper their initial *PS* cannot be higher than 500. However, in case of SapsDE *PS* may increase over 500 during run.

Four among seven DE variants that are not followed by capital letters A or B are the original ones, in which all control parameter settings are kept the same as suggested in the source papers. Capital letters A and B after FDSADE and SapsDE indicate that, in order to verify the selection from the source paper, some modification to the *PS* bounds is introduced. In FDSADE the upper population size suggested in [136] is very small, hence two other variants of *PS* bounds are tested. As in SapsDE there is no upper limit on the population size [149], a variant with such limit is added. Note that, like with the lower *PS* bound in the original SapsDE, the introduced (in SapsDE-A) upper bound may be exceeded – in such case the population reduction strategy is applied.

The choice of DE algorithms with variable population size is motivated as follows. In Section 3.4 it was discussed that algorithms with flexible population sizes could be classified into four groups. The algorithms from the first group, based on the self-adaptation of *PS* at individual level, are not tested here as, according to the previous studies [13,134,135] they do not outperform the basic DE approach. From the second group, with algorithms that reduce the population size during run, jDElscop is chosen, as it is the last variant published in peer-reviewed journal (paper [169] has been published very recently, during second round of reviews of this study). From the third group, that relates the variation of *PS* to the fitness diversity in the population or the function complexity, FDSADE is chosen, as the second algorithm [107] introduces adaptation of control parameters of only the basic DE variant and leads to exceptionally high population sizes. Within the fourth group there are few algorithms based on the idea that the modification of the population size should depend on a number of improvements achieved during recent generations. Two of them are tested, as they follow the contradictory assumptions (see discussion in Section 3.4): SapsDE, in which if there is no improvement for some short time, the *PS* is decreased, and ATPS-JADE, in which if there is no improvement, the *PS* is increased.

5. Results

The presentation of the results is divided into five parts. Sections 5.1–5.3 are devoted solely to the results obtained after maximum number of function calls (as defined for CEC2005 and CEC2011 competitions in [130,30]) is used. In Section 5.1 the impact of the population size on the performance of DE algorithms with fixed population size is discussed. Each among 10 DE algorithms defined in Section 4.2 is tested with various population sizes, specified in Section 4.3, on sets of problems defined in Section 4.1. The aim of this section is to find some general rules on setting the population size in DE algorithms, and answer the questions “how large *PS* should be” and “should *PS* depend on the dimensionality of the problem to be solved”.

In Section 5.2 the performances of DE algorithms with variable population size are compared. As such comparison has been lacking in DE literature so far, this section should help choosing the most promising schemes of DE population size adaptation, answering the question “what concepts the population size adaptation should follow”.

In Section 5.3 all tested DE variants (note that in this study by “DE variant” it is understood the specific DE algorithm with the particular population size setting) are compared together, in order to find: 1. whether DE algorithms with variable *PS* are competitive to the classical ones that have *PS* fixed, 2. how much the ranking of DE algorithms depend on the chosen population size setting, and 3. can the same *PS* be recommended for DE algorithms that share some important features. In other words, this section address the questions “should *PS* be fixed or modified during run” and “how important is *PS* for the performance of DE algorithms”.

In Sections 5.1–5.3 first the results for low-, then higher-dimensional CEC2005 functions are discussed, and finally the conclusions are verified on the real-world CEC2011 problems.

In Section 5.4 the question “how to relate *PS* with the number of allowed function calls” is addressed. This section is based only on CEC2011 problems and 50-dimensional versions of CEC2005 benchmarks. The similar methodology as the one used in Sections 5.1–5.3 is applied, but DE variants are compared according to the results obtained after only 10,000 and 50,000 (in case of CEC2011) or 10,000 and 100,000 (in case of 50-dimensional CEC2005 problems) function calls.

Finally, in Section 5.5 the main results from this paper are reconsidered from the other point of view, by asking not about the averaged performance, but putting attention to question for how many problems each particular DE variant outperform all the others.

5.1. *Differential Evolution algorithms with fixed population size*

The mean performances, accompanied with respective standard deviations, of 10 DE algorithms applied with various fixed population sizes on all considered problems after maximum number of function calls is used are given in Supplementary Tables 1–5. For each DE algorithm separately the best *PS* setting for particular problem is ranked 1, and the worst *PS* setting is ranked 5 (for 2-, 10- and 50-dimensional CEC2005 problems), 7 (for 30-dimensional CEC2005 problems) or 10 (for CEC2011 problems). The averaged over 25 CEC2005 problems with the same dimensionality, or over all 22 CEC2011 real-world problems, rankings of *PS* settings for each algorithm are given in Table 1. The population sizes that achieved the lowest average ranking for particular algorithm are **bolded**. If the difference between the performance achieved with the particular and the best (bolded) *PS* setting is statistically significant at 0.05 significance level according to Holm's procedure, the average ranking of such *PS* is given in *italic*. To generalize results obtained for various algorithms, the averaged over 10 tested DE methods ranking of *PS* settings is given in “mean rank” column in Table 1. Finally, the column “best choice” provides the number of algorithms for which the particular *PS* setting led to the best performance.

To get some knowledge whether the population size setting depend, as one may expect, on the problem difficulty (difficulty is understood intuitively here, without getting into details how to measure it, consult [24,89]) the similar analysis is repeated separately for “simple” unimodal CEC2005 problems F1–F5 (Table 2) and “difficult” CEC2005 hybrid composition functions F15–F25 (Table 3). Note, however, that the global optima of some higher-dimensional unimodal CEC2005 problems (e.g. F3, F5) are rarely

Table 1
Averaged ranking of population sizes of each algorithm when the maximum number of allowed function calls is used. The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	2-Dimensional CEC2005 problems											
3*d*	*5.00*	*4.60*	*4.80*	*4.60*	*4.92*	*4.64*	*4.38*	*4.44*	*4.48*	*4.92*	**4.68**	**0**
5*d*	*3.96*	*3.54*	*3.70*	*3.52*	*3.56*	*3.62*	*3.72*	*3.70*	*3.56*	*3.44*	**3.63**	**0**
10*d*	*2.72*	2.78	*2.72*	2.62	2.62	*2.78*	*2.88*	*2.84*	*2.84*	2.66	**2.75**	**0**
50	1.78	2.10	2.04	**2.12**	**1.94**	2.08	2.18	2.16	2.20	**1.98**	**2.06**	**3**
100	**1.54**	**1.98**	**1.74**	2.14	1.96	**1.88**	**1.84**	**1.86**	**1.92**	2.00	**1.89**	**7**
	10-Dimensional CEC2005 problems											
1*d*	*4.92*	*3.92*	*5.00*	*4.44*	*4.60*	*4.56*	*4.56*	*4.68*	*4.56*	*4.44*	**4.57**	**0**
2*d*	*3.32*	2.82	*3.72*	3.14	*3.32*	*3.58*	*3.38*	*3.54*	*3.62*	*3.40*	**3.38**	**0**
3*d*	2.48	2.86	*2.80*	2.58	2.68	*2.76*	*2.88*	*3.14*	2.70	2.56	**2.74**	**0**
5*d*=50	**2.12**	2.76	1.86	2.54	**2.12**	2.30	2.18	1.88	**1.98**	**1.90**	**2.16**	**4**
10*d*=100	2.16	**2.64**	**1.62**	**2.30**	2.28	**1.80**	**2.00**	**1.76**	2.14	2.70	**2.14**	**6**
	30-Dimensional CEC2005 problems											
1*d*	*4.82*	4.56	*6.02*	*5.64*	*5.28*	*5.60*	*5.42*	*5.70*	*5.56*	*4.48*	**5.31**	**0**
50	3.60	4.04	4.18	4.50	3.96	4.56	*4.96*	*5.00*	*4.80*	3.08	**4.27**	**0**
2*d*	3.24	3.68	4.32	4.18	3.56	3.94	*4.46*	4.18	4.16	3.12	**3.88**	**0**
3*d*	**3.16**	**3.38**	3.78	3.26	3.84	3.76	**3.00**	3.64	3.68	**3.00**	**3.45**	**4**
100	3.36	3.68	3.34	**3.14**	**3.24**	3.46	3.04	3.18	**2.84**	3.16	**3.24**	**3**
5*d*	3.88	4.36	3.22	3.32	3.86	**3.18**	3.56	**3.06**	3.10	*4.80*	**3.63**	**2**
10*d*	*5.94*	4.30	**3.14**	3.96	4.26	3.50	3.56	3.24	3.86	*6.36*	**4.21**	**1**
	50-Dimensional CEC2005 problems											
1*d*=50	*3.04*	**2.70**	*4.12*	*3.80*	3.42	*3.76*	*3.70*	*4.02*	*3.92*	2.26	**3.47**	**1**
2*d*=100	**2.00**	2.74	3.06	2.82	**2.62**	2.92	2.88	2.92	2.86	**2.02**	**2.68**	**3**
3*d*	2.20	2.80	**2.50**	**2.46**	2.76	**2.52**	**2.68**	2.76	**2.54**	2.50	**2.57**	**5**
5*d*	*3.34*	3.30	2.54	2.74	2.96	2.72	2.70	**2.64**	2.58	*3.48*	**2.90**	**1**
10*d*	*4.42*	3.46	2.78	3.18	3.24	3.08	3.04	2.66	3.10	*4.74*	**3.37**	**0**
	CEC2011 problems											
1*d*	*5.43*	5.89	*7.61*	6.25	6.09	6.52	5.48	6.39	6.34	5.64	**6.16**	**0**
2*d*	*6.16*	5.61	*5.84*	5.70	5.36	6.32	5.34	5.25	4.70	5.59	**5.59**	**0**
3*d*	*6.34*	5.39	5.25	5.61	5.41	5.30	5.27	5.02	**4.20**	5.77	**5.36**	**1**
5*d*	*6.98*	5.95	4.23	5.20	5.16	5.02	6.09	5.20	5.52	6.86	**5.62**	**0**
10*d*	*7.39*	*6.70*	**3.36**	5.39	5.91	5.23	*6.95*	5.39	*6.57*	7.86	**6.08**	**1**
20	*6.25*	6.30	*8.75*	*7.02*	6.77	*7.14*	*7.00*	*7.34*	*7.16*	5.59	**6.93**	**0**
30	4.11	5.45	*6.23*	5.66	5.61	5.75	5.68	6.34	5.98	3.86	**5.47**	**0**
50	**3.34**	4.61	*5.86*	**4.66**	**4.64**	4.68	4.41	5.07	4.89	**2.91**	**4.51**	**4**
100	3.52	**4.16**	4.41	4.70	5.05	4.55	4.59	4.61	4.70	3.82	**4.41**	**1**
200	*5.48*	4.93	3.45	4.80	5.00	**4.50**	**4.18**	**4.39**	4.93	7.09	**4.88**	**3**

found by DE algorithms, hence such problems cannot be considered trivial.

For 2-dimensional CEC2005 problems (see Table 1) the worst results are achieved when the smallest population size, equal to $3d$ ($PS=1d$ and $2d$ are skipped for 2-dimensional problems, see Section 4.3) is used. For so low-dimensional problems the larger the population, the better results, and relating PS to problem dimensionality is not recommended. The averaged (over all 10 algorithms) rankings of different population sizes vary noticeably, from 4.68 ($PS=3d$) to 1.89 ($PS=100$) – see the column "mean rank" in Table 1. However, not always the best results were achieved with the largest tested population size – in case of 3 algorithms (SADE, SspDE and CoBiDE) $PS=50$ turns out the slightly better choice than $PS=100$. The differences between results achieved with PS set to 100 and 50 individuals are not statistically significant, but the differences between performances obtained with $PS=100$ and population sizes related to the dimensionality of the problem almost always are.

The impact of the population size on the performance of DE on 2-dimensional unimodal problems (see Table 2) is low, only $PS=3d$ choice is to be avoided. On the contrary, for the most difficult hybrid composition functions $PS=100$ is always the best choice (see Table 3). However, remember that these results are obtained when the maximum number of function calls is used and convergence speed is not considered (convergence speed will be adressed in section 5.4).

For 10-dimensional problems again either $PS=100$, or $PS=50$ individuals turns out the best choice, but the differences between the results achieved with different population sizes have been slightly mitigated comparing to the ones observed for 2-dimensional problems. Although more frequently than in case of 2-dimensional problems the population size equal 50 individuals turns out the best choice, the differences between results obtained with PS fixed to 100 or 50 are again not statistically significant and, with the exception of CoBiDE, truly small.

If the performances of different population size settings are tested on unimodal or the most difficult 10-dimensional problems (Tables 2 and 3), $PS=100$ turns out the best choice. Only in case of CoBiDE smaller population size is suggested for hybrid composition functions.

Table 2
Averaged ranking of population sizes for CEC2005 unimodal problems (F1–F5) when the maximum number of allowed function calls is used. The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	2-Dimensional CEC2005 problems											
3*d*	*5.00*	3.40	*5.00*	3.80	*4.60*	3.40	**3.00**	**3.00**	**3.00**	*4.60*	**3.73**	**4**
5*d*	*4.00*	**2.90**	3.10	**2.80**	**2.60**	**2.90**	**3.00**	**3.00**	**3.00**	**2.60**	**2.99**	**7**
10*d*	**2.00**	**2.90**	**2.30**	**2.80**	**2.60**	**2.90**	**3.00**	**3.00**	**3.00**	**2.60**	**2.71**	**9**
50	**2.00**	**2.90**	**2.30**	**2.80**	**2.60**	**2.90**	**3.00**	**3.00**	**3.00**	**2.60**	**2.71**	**9**
100	**2.00**	**2.90**	**2.30**	**2.80**	**2.60**	**2.90**	**3.00**	**3.00**	**3.00**	**2.60**	**2.71**	**9**
	10-Dimensional CEC2005 problems											
1*d*	*5.00*	*3.80*	*5.00*	*4.20*	*4.20*	*3.80*	3.60	*3.80*	*3.80*	3.40	**4.06**	**0**
2*d*	*3.20*	3.10	2.80	3.00	3.00	3.10	2.90	3.10	3.10	3.20	**3.05**	**0**
3*d*	*3.60*	2.90	2.60	2.80	2.80	2.90	**2.60**	**2.70**	**2.70**	**2.80**	**2.84**	**4**
5*d*=50	1.80	2.70	**2.30**	2.60	2.60	**2.60**	3.30	**2.70**	**2.70**	**2.80**	**2.61**	**5**
10*d*=100	**1.40**	**2.50**	**2.30**	**2.40**	**2.40**	**2.60**	**2.60**	**2.70**	**2.70**	**2.80**	**2.44**	**10**
	30-Dimensional CEC2005 problems											
1*d*	4.70	*5.00*	*5.60*	*6.20*	*5.80*	*5.80*	*4.70*	*5.00*	*5.00*	*4.50*	**5.26**	**0**
50	3.50	*4.40*	*4.40*	*5.00*	*4.80*	*4.40*	*4.50*	4.30	*5.40*	3.90	**4.36**	**0**
2*d*	3.70	*4.40*	*4.00*	*4.20*	3.80	*4.20*	*4.80*	*4.70*	*4.40*	3.30	**4.12**	**0**
3*d*	**3.30**	*4.00*	*4.40*	3.40	3.60	*4.00*	*4.10*	3.80	3.70	3.30	**3.77**	**1**
100	3.70	*4.20*	*3.80*	3.20	*4.00*	*3.60*	*4.10*	4.00	3.50	**2.70**	**3.70**	**1**
5*d*	3.50	*3.80*	3.20	**2.40**	3.40	*3.60*	3.20	3.20	**3.00**	*4.10*	**3.38**	**2**
10*d*	*5.60*	**2.20**	**2.60**	3.60	**2.60**	**2.40**	**2.60**	**3.00**	**3.00**	*6.20*	**3.42**	**7**
	50-Dimensional CEC2005 problems											
1d=50	2.80	3.10	*3.60*	*4.00*	3.70	*4.00*	*3.60*	*3.60*	*3.60*	**1.80**	**3.37**	**1**
2d=100	2.40	2.90	3.00	*3.20*	2.90	3.20	*3.40*	*3.40*	*3.80*	2.20	**3.04**	**0**
3d	**2.00**	3.00	3.00	**2.20**	**2.70**	**2.40**	*3.20*	*3.20*	*3.20*	2.60	**2.75**	**4**
5d	*3.20*	3.20	3.00	2.60	**2.70**	2.80	*3.00*	*3.00*	*2.60*	*3.40*	**2.95**	**1**
10d	*4.60*	**2.80**	**2.40**	3.00	3.00	2.60	**1.80**	**1.80**	**1.80**	*5.00*	**2.88**	**5**

Table 3
Averaged ranking of population sizes for CEC2005 composition functions (F15–F25) when the maximum number of allowed function calls is used. The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	2-Dimensional CEC2005 problems											
3*d*	*5.00*	*5.00*	*4.55*	*4.82*	*5.00*	*4.91*	*4.91*	*4.82*	*4.91*	*5.00*	**4.89**	**0**
5*d*	*3.91*	*3.86*	*3.73*	*3.91*	*3.86*	*4.09*	*3.91*	*4.00*	*4.00*	*3.73*	**3.90**	**0**
10*d*	*3.00*	*3.86*	*3.09*	*2.91*	*2.86*	*2.81*	*3.09*	*3.00*	*3.00*	*3.00*	**3.06**	**0**
50	1.82	1.95	2.14	1.91	1.86	1.86	*1.95*	*2.00*	*1.95*	1.82	**1.93**	**0**
100	**1.27**	**1.32**	**1.50**	**1.45**	**1.41**	**1.32**	**1.14**	**1.18**	**1.14**	**1.45**	**1.32**	**10**
	10-Dimensional CEC2005 problems											
1*d*	*5.00*	*4.36*	*5.00*	*5.00*	*5.00*	*5.00*	*5.00*	*5.00*	*5.00*	*4.91*	**4.93**	**0**
2*d*	*3.45*	2.91	*4.00*	*3.73*	*3.73*	*4.00*	*3.64*	*3.55*	*3.91*	*3.64*	**3.66**	**0**
3*d*	2.09	*3.18*	*2.91*	2.45	*2.64*	*2.50*	*3.27*	*3.45*	*2.91*	*2.64*	**2.80**	**0**
5*d*=50	2.41	2.45	1.64	2.18	1.91	*2.18*	1.73	1.82	1.82	**1.73**	**1.99**	**1**
10*d*=100	**2.05**	**2.09**	**1.45**	**1.64**	**1.73**	**1.32**	**1.36**	**1.18**	**1.36**	2.09	**1.63**	**9**
	30-Dimensional CEC2005 problems											
1*d*	*5.73*	*5.55*	*6.59*	*6.09*	*6.09*	*6.45*	*6.18*	*5.55*	*6.45*	*4.77*	**5.89**	**0**
50	*4.41*	*4.50*	*4.23*	*5.14*	3.86	*5.32*	*5.59*	*5.91*	*5.64*	**3.00**	**4.66**	**1**
2*d*	3.41	3.82	4.00	*4.77*	3.68	3.64	*4.69*	*4.55*	*4.14*	3.64	**4.32**	**0**
3*d*	**2.68**	**2.82**	3.50	3.50	3.73	3.68	3.14	*4.00*	*4.09*	3.18	**3.36**	**2**
100	2.86	3.09	3.86	3.05	**2.77**	3.50	**2.59**	3.05	2.82	**3.00**	**3.09**	**3**
5*d*	3.41	4.09	3.05	2.91	3.73	**2.59**	3.18	2.59	**2.18**	*4.50*	**3.34**	**2**
10*d*	*5.50*	4.14	**2.77**	**2.55**	4.14	2.82	2.64	**2.36**	2.68	*5.91*	**3.65**	**3**
	50-Dimensional CEC2005 problems											
1*d*=50	*4.09*	3.09	*4.64*	*4.45*	*3.91*	*4.55*	*4.36*	*5.00*	*4.64*	2.64	**4.14**	**0**
2*d*=100	2.09	3.00	*3.23*	*3.41*	3.09	3.00	*3.32*	*3.18*	2.86	**2.36**	**2.95**	**1**
3*d*	**2.00**	**2.64**	2.41	2.50	2.68	2.64	2.50	*2.82*	2.59	**2.36**	**2.51**	**3**
5*d*	*2.95*	2.82	**2.23**	**2.32**	2.68	2.45	2.45	2.23	**2.05**	3.18	**2.54**	**3**
10*d*	*3.86*	3.18	2.50	**2.32**	**2.64**	**2.36**	**2.36**	**1.77**	2.86	*4.45*	**2.83**	**5**

Overall, for low-dimensional artificial benchmark problems the population size set to 100 is the best and the safest choice, what confirms the trend of setting it for various DE algorithms that is observed in recent years. Among 10 tested DE algorithms, only CoBiDE often performs better with smaller population size. This observation, however, is not valid for higher dimensional problems.

When 30-dimensional problems are considered the best population size settings diversify, and vary between 3*d* and 5*d*; however, even 10*d* may be the best choice in case of MDE_pBX. Although one may note that for 7 out of 10 DE algorithms the population size equal 90 (i.e. 3*d*) or 100 individuals is still the best choice, and the average rankings of MDE_pBX or Rcr-JADE with *PS* equal 100, 5*d* and 10*d* differ only marginally (see Table 1), the important observations are made for the simplest (Table 2) and the most difficult problems (Table 3). Contrary to what one may expect, in case of unimodal problems the largest *PS* (equal to 10*d*) leads to the best results for 7 out of 10 DE algorithms, and *PS* equal to 90–100 individuals may only be recommended for CDE and CoBiDE. The differences between results achieved with $PS=10d$ and 100 individuals are frequently statistically significant, although the sample is small. In case of hybrid composition functions the picture is blurred and, depending on the algorithm, the best *PS* choice may vary from 3*d* to 10*d*. For some algorithms (CDE, CoBiDE, Rcr-JADE or JADE-EIG) differences between results achieved with *PS* varying from 3*d* to 10*d* are large and statistically significant.

For 50-dimensional problems, 5 out of 10 tested DE algorithms perform best with *PS* set to 3*d* (i.e. 150 individuals); $PS=100$ (i.e. 2*d*) turns out the best choice for 3 algorithms (CDE, SADE and CoBiDE). Although much different population sizes turned out best for EPSDE ($PS=50$) and Rcr-JADE ($PS=5d$), if this two algorithms are applied with $PS=3d$ or 100, their performances deteriorates only marginally.

Surprisingly, the choice of *PS* turns out very difficult for 50-dimensional unimodal problems (see Table 2). Half of tested methods require the largest considered population size (performance of MDE_pBX and all DE algorithms based on JADE much deteriorates when *PS* is reduced to, or below, 5*d*), but 4 other algorithms perform best with $PS=3d$, and CoBiDE even 1*d*. The choice of $PS=100$ or 50 is, with the exception of CoBiDE, not recommended. Comparison of the results achieved for 30- and 50-dimensional unimodal problems show how sensitive the choice of the population size to the problem dimensionality may be.

For 50-dimensional hybrid composition functions (see Table 3) the best *PS* settings vary between 3*d* and 10*d*, depending on the algorithm. Although $PS=10d$ seems the best choice for 5 out of 10 DE algorithms, the differences between results achieved with $PS=10d$ than 5*d* are not statistically significant. For many algorithms (MDE_pBX, SspDE, SADE, RB-SADE, AdapSS-JADE) any choice between 3*d* and 10*d* is almost equally good. The difference between the best *PS* and the *PS* composed of 100 individuals is in many cases statistically significant, hence setting *PS* to 100 cannot be recommended.

To summarize, based on CEC2005 artificial benchmarks the setting of $PS=100$ is recommended as the first choice for low-dimensional problems; not much poorer results are achieved with *PS* fixed to 50 individuals. However, such population sizes may be too small for 30-dimensional problems, and are too small for 50-dimensional ones. In case of both 30- and 50-dimensional problems using $PS=50$ may only be recommended to CoBiDE. Although it is difficult to linearly relate the population size to the problem dimensionality, the relation $PS=3d$, or 5*d*, is the safest for moderately high-dimensional problems. Population size set to 10*d* cannot be recommended, as it hampers the search of many DE algorithms. Even if for some approaches (i.e. MDE_pBX) so large *PS* is useful when the problem is very hard and multimodal, such algorithms achieve similar performances with $PS=5d$, that is much safer choice.

Are these conclusions confirmed on real-world CEC2011 problems? When all 22 CEC2011 problems (with dimensionalities varying from 1 to 216) are considered (see Table 1), the best results for 7 out of 10 DE algorithms are obtained with population sizes fixed to either 50, 100 or 200 individuals. The average ranking is the lowest for $PS=100$, but most frequently the best choice is $PS=50$. Only MDE_pBX and Rcr-JADE perform better when the population size is related to the dimensionality of the problem, but each of them require different relation – for MDE_pBX $PS=10d$, and for Rcr-JADE $PS=3d$ is suggested. One may note, hence, that the impact of *PS* setting on the performance of various DE algorithms tested on real-world problems is hard to generalize. It is only clear that the population sizes lower than 50 individuals (considering problem dimensionality-independent settings) or 3*d* (for dimensionality-dependent ones) are not recommended; the differences between the results obtained with so low population sizes and the best *PS* settings are frequently statistically significant (see Table 1).

It is, however, important to put some attention to the results obtained when low-dimensional problems (those with dimensionality below 30, but excluding two rather specific 1-dimensional ones; there are 9 such problems in CEC2011 set), or relatively high-dimensional ones (those with dimensionality above 40, there are 8 such problems in CEC2011) are considered separately (see Table 4). In case of real-world low-dimensional problems the population size set to either 50 or 100 is always the best choice. This confirms that DE population size for low-dimensional problems should not be related to the problem dimensionality. As the average rankings of DE variants with $PS=50$ and 100 differ only marginally, and for none algorithm this difference is statistically significant, the results obtained for low-dimensional real-world problems do not contradict the findings based on low-dimensional artificial CEC2005 benchmarks.

In case of relatively high-dimensional CEC2011 problems the population size should be much larger; for 6 out of 10 algorithms the best results are obtained with $PS=200$ (Table 4). However, so large and fixed population size is highly unfavorable for CoBiDE and CDE, and at least disputable for MDE_pBX, hence this cannot be a universal choice. The results obtained with population size set to 100 individuals, or related to the problem dimensionality by $PS=2–3d$ are frequently satisfactory, but again for some algorithms such population sizes lead to poor performance. A few algorithms (MDE_pBX and RB-SADE) perform best when the population size is related to the problem dimensionality by $PS=5–10d$, what is clearly not advised for most others. As for higher-dimensional problems $d>40$, the population sizes noticeably higher than 100 individuals could be suggested for most algorithms, but CDE and CoBiDE require $PS=100$ or 50, and perform very poorly when $PS>100$. It must be hence concluded that for high-dimensional real-world problems there is no universal choice of population size that could be safely recommended.

The general findings regarding the choice of the population size based on both artificial CEC2005 and real-world CEC2011 problems are not as clear as one wish to obtain, but may be summarized as follows: 1. in case of low-dimensional problems (dimensionality up to 30) the population size should be fixed to 50–100 individuals, and do not need to be related to the problem dimensionality; 2. population sizes lower than 50 are not recommended; 3. population size should be higher than 100, and may be related to problem dimensionality ($PS=3d$ or 5*d* is suggested as a first choice) in case of relatively high-dimensional artificial problems; 4. setting the population size for real-world higher-dimensional problems (those with dimensionality over 40)

Table 4
Averaged ranking of population sizes when lower- and higher-dimensional CEC2011 problems are considered separately (based on the results obtained when the maximum number of allowed function calls is used). As lower-dimensional problems are considered those with dimensionality below 30, excluding 1-dimensional ones (P1, P7, P8, P10, P11.3, P11.4, P11.5, P12, P13). As higher-dimensional problems are considered those with dimensionality 40 and higher (P9, P11.1, P11.2, P11.6, P.11.7, P11.8, P11.9, P11.10). The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	Lower-dimensional CEC2011 problems											
1*d*	*8.00*	*7.78*	*9.22*	*8.22*	7.06	*8.00*	*7.89*	*8.22*	*8.56*	*7.83*	**8.08**	**0**
2*d*	6.67	6.56	*6.89*	6.67	5.94	*7.67*	6.78	6.67	6.44	5.72	**6.60**	**0**
3*d*	5.56	5.89	6.33	6.33	5.61	5.61	6.28	5.89	4.33	4.39	**5.62**	**0**
5*d*	5.00	4.61	4.39	4.78	4.67	5.33	4.39	4.89	4.44	4.50	**4.70**	**0**
10*d*	4.89	5.00	3.94	5.00	5.39	4.94	4.61	4.89	5.67	6.06	**5.04**	**0**
20	7.33	6.11	*8.11*	6.22	7.17	6.61	6.94	6.44	7.00	*6.94*	**6.89**	**0**
30	4.78	5.06	5.06	4.78	5.33	5.44	5.17	5.89	5.00	4.61	**5.11**	**0**
50	4.00	4.00	4.28	**3.89**	**4.17**	**3.61**	**3.39**	4.44	**3.89**	**2.94**	**3.86**	**6**
100	**3.78**	**3.89**	**3.28**	4.22	4.61	3.72	4.61	**3.44**	**3.89**	4.06	**3.95**	**5**
200	5.00	6.11	3.50	4.89	5.06	4.06	4.94	4.22	5.78	*7.94*	**5.15**	**0**
	Higher-dimensional CEC2011 problems											
1*d*	3.63	4.88	5.63	6.00	6.25	6.00	4.13	5.38	4.75	3.88	**5.05**	**0**
2*d*	*5.88*	4.25	3.88	5.13	4.63	5.13	4.00	4.00	3.00	*6.13*	**4.60**	**0**
3*d*	*7.25*	4.63	4.13	4.13	4.50	4.38	3.38	3.50	**2.88**	*7.38*	**4.62**	**1**
5*d*	*8.63*	*6.38*	3.63	4.13	4.63	**3.50**	*7.38*	4.00	5.63	*9.13*	**5.70**	**1**
10*d*	*9.25*	*7.38*	**2.75**	4.00	4.63	**3.50**	*8.50*	4.13	6.00	*9.50*	**5.96**	**2**
20	*6.75*	*8.63*	*10.00*	*9.38*	*8.00*	*9.13*	*8.63*	*9.75*	*9.13*	4.38	**8.38**	**0**
30	4.25	*7.25*	*8.50*	*8.00*	7.50	*7.50*	*7.13*	*8.50*	*8.38*	3.13	**7.01**	**0**
50	2.13	5.63	*7.13*	6.13	5.88	6.63	*6.13*	*7.25*	*7.25*	2.63	**5.68**	**0**
100	**2.00**	3.50	5.50	4.63	5.13	5.50	4.00	5.38	5.13	**2.50**	**4.33**	**2**
200	5.25	**2.50**	3.88	**3.50**	**3.88**	3.75	**1.75**	**3.13**	**2.88**	*6.38*	**3.69**	**6**

is difficult and have large impact on the performance; setting *PS*=200 is the first suggestion, but for many algorithms population sizes set to some value between [50,10*d*] (related to problem dimensionality, or not) may perform much better.

Algorithms that share large similarities often require related, but not the same population sizes. For example (consult Table 1), three JADE-based algorithms perform best with almost the same population sizes for CEC2005 problems, but not for CEC2011 ones. RB-SADE performs best with a bit higher population size than SADE. CoBiDE almost always require lower population size than JADE-EIG. Even if these differences are not statistically significant, they are consistent for all performed comparisons.

5.2. *Differential Evolution algorithms with flexible population size*

The performances of 7 tested DE variants with flexible population size achieved for every problem after maximum number of function calls is used are given at the end of each Supplementary Tables 1–5. Such DE variants are ranked from the best (rank 1) to the worst (rank 7) for each problem. The rankings averaged over all considered problems are given in Table 5.

The dependence of the results on the problem dimensionality is observed. For 2-dimensional CEC2005 problems the variant FDSADE-A perform best, followed by ATPS-JADE. For 10-dimensional problems jDElscop clearly outperform all competitors. However for higher-dimensional CEC2005 and for CEC2011 problems ATPS-JADE is definitely the best algorithm with flexible population size; for such problems the differences between the results achieved by ATPS-JADE and the majority of other variants are statistically significant.

Overall ATPS-JADE is the most promising approach. It is ranked first on higher-dimensional or real-world problems and second on 2-dimensional CEC2005 benchmarks. By surprise, it performs poorly for 10-dimensional CEC2005 problems. From Supplementary Table 2 one may find that results achieved by ATPS-JADE are disappointing for 10-dimensional hybrid composition functions.

The upper limit of the population size introduced in SapsDE-A algorithm turns out not needed for low-dimensional problems, but plays some role for higher-dimensional ones, for which SapsDE-A outperforms SapsDE. As discussed in Section 3.4, the concepts of SapsDE and ATPS-JADE share many similarities, but also have some important differences. Based on the performed experiments, ATPS-JADE must be praised as the better of the two approaches. However, SapsDE-A turns out the second-best DE algorithm with flexible population size for higher-dimensional or real-world problems, and the differences between the two are not statistically significant.

The performance of the fitness diversity-based population size adaptation method (FDSADE variants) depends largely on the *PS*

Table 5
Averaged ranking of algorithms with variable population size when the maximum number of allowed function calls is used. The ranking of particular algorithm is given in *italic* if the difference between the results achieved with this method and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Algorithm	2-Dimensional CEC2005	10-Dimensional CEC2005	30-Dimensional CEC2005	50-Dimensional CEC2005	CEC2011	**Mean rank**	**Best choice**
jDElscop	*5.16*	**2.78**	4.00	*4.30*	*4.27*	**4.10**	**1**
FDSADE	3.90	*4.80*	*5.36*	*5.04*	*4.73*	**4.77**	**0**
FDSADE-A	**2.78**	3.58	3.88	*3.80*	*4.45*	**3.70**	**1**
FDSADE-B	*5.78*	4.08	*4.92*	*5.72*	*5.23*	**5.15**	**0**
ATPS-JADE	3.20	*4.52*	**2.76**	**2.28**	**2.59**	**3.07**	**3**
SapsDE	3.62	3.94	3.70	*3.82*	3.50	**3.72**	**0**
SapsDE-A	3.56	*4.30*	3.38	3.04	3.23	**3.50**	**0**

bounds. Introduced in this paper FDSADE-A variant perform well for low-dimensional problems, but other two, including the original FDSADE, cannot be praised. Although the idea of using fitness-diversity to vary the population size is encouraging, the more robust fitness-diversity measures and population adaptation rules are probably needed, and they should be coupled with better DE scheme than the one used in FDSADE.

jDElscop, with its simple idea of gradual population size reduction, turns out the best for 10-dimensional CEC2005 problems, but otherwise is not well ranked. Its performance heavily depends on the problem. For example, from Supplementary Table 4 one may find that the ranking of jDElscop vary seriously for 50-dimensional hybrid composition functions (for 4 out of 11 such problems jDElscop is the best among DE variants with variable population sizes, for 4 others – the worst). jDElscop performance on real-world problems is similarly diversified (see Supplementary Table 5). Interestingly, this algorithm was also ranked poorly when compared with various metaheuristics on minimization of CEC2005 functions in [113], but turned out the best when the maximum of such functions were searched. Analysis given in [34] also suggest that the ranking of jDElscop very heavily depend on the way the competition is organized. The reason of such diversified performance of jDElscop needs to be verified in the future, but the basic idea of starting from the large population and constantly diminishing it during run is surely worth attention, due to both extreme simplicity, theoretical justification (it facilitates exploration during early part of the run and forces exploitation at the end) and good performance on some kinds of problems.

5.3. *Differential Evolution variants with fixed and flexible population sizes – a comparison*

In Table 6 all considered DE variants are ordered according to their average rankings from the best (the one with the lowest average rank) to the poorest (the one with the highest average rank), when all algorithms are compared together after maximum number of function calls is used. In case of 2-, 10- and 50-dimensional CEC2005 problems 57 variants are compared, for 30-dimensional CEC2005 problems number of variants equal 77, and for CEC2011 ones – 107 (for the reasons see Section 4.3). The average rankings given in Table 6 are obtained as follows: first each considered DE variant is ranked from 1st to 57th (77th, or 107th) on every among 25 (or 22, in case of CEC2011) problems, and then its ranking is averaged over all problems. When “mean rank” of particular variant is given in *italic* it means that the differences between the results achieved by such variant and the best one (ranked 1) are statistically significant at 0.05 significance level according to Holm's procedure.

Note that the order of algorithms in rankings given in Supplementary Tables 1–5 and in Table 6 may differ. If among two variants that were compared with themselves in Supplementary Tables 1–5 the variant A was better than B, it does not exclude the possibility that when all 57 (77 or 107) variants are compared, variant B would gain lower rank. Imagine that among 5, 7 or 10 variants compared in Supplementary Tables 1–5 variants A and B perform equally for all but two problems. If A was ranked 1st and B 3rd on problem I, but B was ranked 1st and A 2nd on problem II, the average ranking of A was lower (better) than B. However, when all 57 (77 or 107) variants of all tested algorithms are compared together, it may happen that for example on problem I variant A is still two ranks better than B, but on problem II the performance of many variants fall between B and A. As a result in Table 6 the average rank of B would be lower (better) than the average rank of A. In practice such cases are rare and have no impact on the general conclusions from the paper, but may be spotted by the cautious reader.

The main general conclusions from Table 6 are as follows:

1. Differences between the results obtained by at least 50% of variants and the best one are not statistically significant. For CEC2005 problems (irrespective of the dimensionality) a number of best variants share almost the same average ranking. On the other hand, large diversity of average rankings is observed among the poorest variants. The results on real-world problems show more diversified performance among the best DE variants, what suggest that non-artificially-created problems are more selective.
2. In case of 2- and 10-dimensional CEC2005 problems the lower the population size is, the poorer performance (when only DE variants with fixed population size are considered). However, such rule is disputable for real-world problems, and not observed for 30- or 50-dimensional CEC2005 benchmarks. This means that the scarcity of possible difference vectors hampers all DE algorithms, but the impact of the profusion of them (within some common-sense limits) cannot be generalized.
3. Population size has high impact on the performance of many DE algorithms. For example, for 2-dimensional CEC2005 problems CoBiDE-50 is the third best variant, but CoBiDE-3*d* is the 5th worst; for real-world problems CoBiDE-50 is the best variant, but CoBiDE-10*d* – the sixth worst. Depending on the population size Rcr-JADE and RB-SADE may be ranked from 2nd and 3rd best to 3rd and 4th worst variants for 10-dimensional CEC2005 benchmarks. For 30-dimensional CEC2005 problems JADE-EIG-100 is the 3rd best variant, but JADE-EIG-1*d* (population composed of 30 individuals) – the 8th worst, out of 77. As $PS=100$ is frequently set, but $PS=30$, or even lower, is also used in some DE papers [8,146], this show how easily the performance of DE algorithms my be bend by setting fixed (of course to the value that best suits the algorithm one promotes) population size for all competing methods.
4. Contrary to the discussion given in the point 3, sometimes the performances of selected (usually JADE-based) algorithms may be almost non-sensitive to the population size, even if it is varied within relatively wide limits. For example, although the large sensitivity of Rcr-JADE to population size has been observed for 10-dimensional CEC2005 benchmarks (see point 3 above), variants of the same Rcr-JADE with population sizes that differ by 50–100 individuals were ranked 1st and 2nd on 30-dimensional, and 2nd and 3rd on 50-dimensional CEC2005 problems, surprisingly showing very little sensitivity to the population size. JADE-EIG and Rcr-JADE with very different *PS* settings tested on real-world problems occupy 13 out of 20 best places, enjoying good performance that is also mildly affected by the population size. This show that some DE variants may be safely applied with various population sizes for many (but not all) problems, if such population sizes are set within a reasonable limit. However, some common *PS* settings outside such “tolerance interval” may knock the performance down (for example, in case of 30-dimensional CEC2005 problems Rcr-JADE with $PS = 1d$ is ranked only 60th).
5. In case of low-dimensional CEC2005 problems 3 best variants are those with fixed and dimensionality-independent population size. However, in case of 30-dimensional CEC2005 benchmarks the variant with $PS=5d$, and in case of 50-dimensional ones – the variant with adaptive population size turned out the best. In case of 50-dimensional problems the best 4 variants, and 8 out of 10 best variants, relates the population size to the problem dimensionality or adapts it during run. This show that for low-dimensional problems the simplest choice by setting *PS* to 100 or 50 is recommended, but in case of higher-dimensional artificial problems the population size should be larger, and either related to the problem dimensionality or adaptive. In case

Table 6
Ranking of all considered DE variants (here variant is understood as the algorithm with the specified population size; e.g. if one algorithm is tested with three population sizes, there are three variants) when the maximum number of allowed function calls is used. Differences between DE variants which "mean ranks" are given in *italic* and the best variant (ranked 1) are statistically significant at 5% significance level according to Holm's procedure.

	2-Dimensional CEC2005		10-Dimensional CEC2005		30-Dimensional CEC2005		50-Dimensional CEC2005		CEC2011	
Ranking	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant
1	15.50	Rcr-JADE-100	14.32	CoBiDE-5*d*=50	21.20	Rcr-JADE-5*d*	15.82	ATPS-JADE	28.30	CoBiDE-50
2	15.64	EPSDE-100	15.72	Rcr-JADE-10*d*=100	22.62	Rcr-JADE-100	15.98	Rcr-JADE-5*d*	30.11	ATPS-DE
3	15.66	CoBiDE-50	16.52	RB-SADE-10*d*=100	24.52	JADE-EIG-100	16.52	Rcr-JADE-3*d*	32.25	Rcr-JADE-200
4	15.66	SADE-50	16.88	jDElscop	25.40	AdapSS-JADE-100	16.90	AdapSS-JADE-5*d*	32.91	Rcr-JADE-100
5	15.82	JADE-EIG-100	17.42	CoBiDE-3*d*	25.50	AdapSS-JADE-3*d*	17.76	Rcr-JADE-2*d*=100	33.39	JADE-EIG-100
6	16.02	RB-SADE-100	18.70	SADE-5*d*=50	25.76	CoBiDE-2*d*	18.06	AdapSS-JADE-3*d*	34.39	CoBiDE-30
7	16.28	CDE-100	19.14	CDE-5*d*=50	26.16	Rcr-JADE-10*d*	18.38	JADE-EIG-3*d*	34.84	JADE-EIG-3*d*
8	16.36	SspDE-50	19.20	JADE-EIG-10*d*=100	26.70	JADE-EIG-5*d*	18.86	AdapSS-JADE-2*d*=100	35.93	JADE-EIG-200
9	16.40	SADE-100	19.60	SADE-10*d*=100	27.24	ATPS-JADE	19.04	JADE-EIG-5*d*	37.16	MDE-pBX-200
10	16.50	EPSDE-50	20.08	SspDE-10*d*=100	27.66	CoBiDE-50	19.94	Rcr-JADE-10*d*	37.57	JADE-EIG-2*d*
11	16.94	SspDE-100	21.28	RB-SADE-5*d*=50	28.26	CoBiDE-3*d*	20.74	JADE-EIG-2*d*=100	37.89	RB-SADE-200
12	16.98	CoBiDE-100	21.34	FDSADE-A	28.30	AdapSS-JADE-5*d*	20.80	RB-SADE-3*d*	37.93	Rcr-JADE-3*d*
13	17.72	FDSADE-A	21.52	Rcr-JADE-5*d*=50	29.28	Rcr-JADE-3*d*	21.08	SapsDE-A	39.43	RB-SADE-100
14	17.74	AdapSS-JADE-100	21.70	JADE-EIG-5*d*=50	29.92	CoBiDE-100	21.38	RB-SADE-5*d*	39.57	Rcr-JADE-5*d*
15	17.82	RB-SADE-50	21.86	CoBiDE-10*d*=100	29.96	RB-SADE-5*d*	22.98	AdapSS-JADE-10*d*	40.02	JADE-EIG-50
16	18.68	CDE-50	21.96	CDE-10*d*=100	30.64	SADE-100	24.34	JADE-EIG-10*d*	40.16	Rcr-JADE-2*d*
17	18.80	JADE-EIG-50	21.98	AdapSS-JADE-10*d*=100	30.90	JADE-EIG-3*d*	24.74	RB-SADE-2*d*=100	41.48	Rcr-JADE-50
18	20.00	ATPS-JADE	22.20	SspDE-5*d*=50	31.00	AdapSS-JADE-10*d*	25.24	CoBiDE-2*d*=100	41.57	JADE-EIG-5*d*
19	20.84	SapsDE-A	22.72	SspDE-3*d*	31.00	SapsDE-A	25.34	MDE_pBX-3*d*	41.66	Rcr-JADE-10*d*
20	21.00	Rcr-JADE-50	22.94	SapsDE	31.94	SADE-2*d*	25.36	SADE-2*d*=100	41.66	SapsDE-A
21	21.46	SspDE-10*d*	23.16	SADE-3*d*	32.04	SapsDE	25.46	CoBiDE-1*d*=50	42.02	MDE_pBX-10*d*
22	21.88	CoBiDE-10*d*	23.54	AdapSS-JADE-5*d*=50	32.26	Rcr-JADE-2*d*	25.52	SADE-3*d*	42.32	SADE-100
23	22.22	AdapSS-JADE-50	24.06	FDSADE-B	32.90	RB-SADE-10*d*	26.02	SapsDE	42.36	RB-SADE-5*d*
24	22.38	SapsDE	24.78	EPSDE-10d=100	33.12	JADE-EIG-10*d*	26.28	MDE_pBX-5*d*	42.98	MDE_pBX-100
25	22.76	MDE_pBX-100	24.78	SapsDE-A	33.34	SADE-3*d*	26.64	RB-SADE-10*d*	43.27	SADE-200
26	23.34	SADE-10*d*	25.12	RB-SADE-3*d*	33.46	MDE_pBX-5*d*	27.12	SADE-5*d*	44.09	RB-SADE-50
27	23.84	FDSADE	25.44	CoBiDE-2*d*	33.78	RB-SADE-100	27.22	AdapSS-JADE-1*d*=50	44.84	RB-SADE-10*d*
28	24.82	EPSDE-10*d*	26.02	MDE_pBX-10*d*=100	35.06	SADE-5*d*	28.28	MDE_pBX-2*d*=100	44.84	SapsDE
29	24.94	MDE_pBX-50	26.64	CDE-3*d*	35.10	RB-SADE-3*d*	29.28	CDE-2*d*=100	45.30	SADE-50
30	27.46	JADE-EIG-10*d*	26.64	EPSDE-5*d*=50	35.14	MDE_pBX-100	*29.68*	MDE_pBX-10*d*	45.52	CoBiDE-100
31	28.58	RB-SADE-10*d*	26.90	FDSADE	35.52	JADE-EIG-2*d*	*29.90*	SspDE-5*d*	45.55	RB-SADE-3*d*
32	28.92	Rcr-JADE-10*d*	27.08	ATPS-JADE	35.78	jDElscop	*30.26*	SADE-10*d*	45.89	EPSDE-100
33	*29.42*	jDElscop	27.46	EPSDE-2*d*	36.70	SADE-50	*30.50*	CoBiDE-3*d*	46.11	AdapSS-JADE-100
34	*29.96*	AdapSS-JADE-10*d*	27.92	MDE_pBX-5*d*=50	36.84	AdapSS-JADE-2*d*	*30.74*	jDElscop	46.11	MDE_pBX-5*d*
35	*30.66*	CDE-10d	28.10	JADE-EIG-3*d*	36.84	MDE_pBX-3*d*	*30.88*	FDSADE-A	46.39	SspDE-100
36	*31.74*	SspDE-5d	*28.52*	SspDE-2*d*	36.90	SspDE-100	*31.64*	Rcr-JADE-1*d*=50	46.70	AdapSS-JADE-50
37	*32.86*	FDSADE-B	*29.06*	SADE-2*d*	36.94	MDE_pBX-10*d*	*32.34*	CDE-3*d*	47.16	AdapSS-JADE-200
38	*32.98*	CoBiDE-5*d*	*29.40*	EPSDE-3*d*	36.94	SspDE-5*d*	*32.48*	SADE-1*d*=50	48.11	SADE-2*d*
39	*32.98*	EPSDE-5*d*	*30.22*	AdapSS-JADE-3*d*	37.60	RB-SADE-2*d*	*32.60*	SspDE-3*d*	48.68	SADE-5*d*
40	*33.18*	MDE_pBX-10*d*	*31.56*	Rcr-JADE-3*d*	38.76	Rcr-JADE-50	*32.64*	RB-SADE-1*d*=50	48.73	JADE-EIG-1*d*
41	*34.88*	JADE-EIG-5*d*	*32.94*	RB-SADE-2*d*	39.10	FDSADE-A	*32.98*	SspDE-10*d*	48.91	SspDE-200
42	*35.26*	SADE-5*d*	*33.88*	JADE-EIG-2*d*	39.32	MDE_pBX-2*d*	*33.56*	JADE-EIG-1*d*=50	50.07	JADE-EIG-10*d*
43	*37.94*	RB-SADE-5*d*	*34.02*	AdapSS-JADE-2*d*	39.50	AdapSS-JADE-50	*33.58*	SspDE-2*d*=100	50.41	SADE-3*d*
44	*38.32*	Rcr-JADE-5*d*	*34.78*	Rcr-JADE-2*d*	39.86	CoBiDE-1*d*	*33.62*	CDE-1*d*=50	50.89	SspDE-50
45	*40.18*	AdapSS-JADE-5*d*	*35.20*	MDE_pBX-3*d*	40.10	CDE-2*d*	*35.56*	EPSDE-3*d*	51.07	JADE-EIG-30
46	*43.38*	SspDE-3*d*	*40.36*	EPSDE-1*d*	40.20	SADE-10*d*	*36.56*	EPSDE-10*d*	51.11	CDE-50
47	*43.90*	Rcr-JADE-3*d*	*40.54*	CoBiDE-1*d*	40.52	JADE-EIG-50	*37.02*	EPSDE-2*d*=100	51.70	CoBiDE-20
48	*44.32*	JADE-EIG-3*d*	*41.94*	MDE_pBX-2*d*	40.62	CDE-50	*37.24*	MDE_pBX-1*d*=50	51.73	MDE_pBX-3*d*
49	*44.96*	AdapSS-JADE-3*d*	*43.84*	SspDE-1*d*	*41.10*	SspDE-3*d*	*37.70*	EPSDE-5*d*	51.82	EPSDE-200
50	*45.10*	EPSDE-3*d*	*44.32*	AdapSS-JADE-1*d*	*41.46*	SspDE-10*d*	*37.98*	FDSADE	52.52	Rcr-JADE-30
51	*46.84*	MDE_pBX-5*d*	*44.56*	JADE-EIG-1*d*	*42.04*	CDE-100	*39.30*	CDE-5*d*	52.93	EPSDE-50
52	*47.18*	RB-SADE-3*d*	*44.60*	SADE-1*d*	*42.48*	RB-SADE-50	*40.06*	SspDE-1*d*=50	53.07	MDE_pBX-50
53	*47.62*	CoBiDE-3*d*	*44.68*	CDE-2*d*	*43.28*	CDE-3*d*	*40.68*	EPSDE-1*d*=50	53.09	CDE-100
54	*48.74*	SADE-3*d*	*45.58*	Rcr-JADE-1*d*	*43.94*	MDE_pBX-50	*42.34*	CoBiDE-5*d*	53.36	Rcr-JADE-1*d*
55	*49.60*	CDE-5*d*	*45.92*	RB-SADE-1*d*	*45.36*	CoBiDE-5*d*	*42.48*	FDSADE-B	53.59	RB-SADE-30
56	*55.88*	MDE_pBX-3*d*	*55.96*	CDE-1*d*	*45.94*	CDE-5*d*	*45.54*	CDE-10*d*	53.80	SADE-10*d*
57	*56.16*	CDE-3*d*	*56.40*	MDE_pBX-1*d*	*46.06*	SspDE-2*d*	*52.06*	CoBiDE-10*d*	54.25	AdapSS-JADE-3*d*

Table 6 (*continued*)

	2-Dimensional CEC2005		10-Dimensional CEC2005		30-Dimensional CEC2005		50-Dimensional CEC2005		CEC2011	
Ranking	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant	Mean ranks	DE variant
58					*46.14*	FDSADE-B			55.02	AdapSS-JADE-30
59					*46.80*	SspDE-50			55.16	RB-SADE-2d
60					*47.44*	Rcr-JADE-1d			55.23	SADE-30
61					*47.54*	EPSDE-3d			55.48	SspDE-5d
62					*47.86*	EPSDE-10d			55.75	SADE-1d
63					*48.18*	EPSDE-100			56.05	MDE_pBX-2d
64					*48.58*	AdapSS-JADE-1d			56.66	CDE-30
65					*48.74*	FDSADE			56.84	SspDE-10d
66					*49.58*	SADE-1d			56.95	SspDE-3d
67					*49.94*	EPSDE-5d			57.39	AdapSS-JADE-2d
68					*50.08*	EPSDE-2d			57.41	AdapSS-JADE-1d
69					*50.72*	RB-SADE-1d			*58.20*	CoBiDE-1d
70					*51.68*	JADE-EIG-1d			*58.61*	SspDE-2d
71					*51.76*	CDE-1d			*58.84*	JADE-EIG-20
72					*52.44*	EPSDE-50			*59.16*	Rcr-JADE-20
73					*55.76*	SspDE-1d			*59.52*	EPSDE-3d
74					*56.64*	MDE_pBX-1d			*59.64*	jDElscop
75					*59.74*	EPSDE-1d			*60.66*	SspDE-30
76					*60.24*	CDE-10d			*60.70*	FDSADE
77					*63.28*	CoBiDE-10d			*60.82*	RB-SADE-1d
78									*60.91*	EPSDE-5d
79									*61.16*	MDE-pBX-30
80									*61.20*	CoBiDE-3d
81									*62.27*	AdapSS-JADE-5d
82									*62.30*	FDSADE-A
83									*62.66*	CoBiDE-2d
84									*62.93*	EPSDE-2d
85									*63.59*	EPSDE-10d
86									*63.80*	AdapSS-JADE-20
87									*64.05*	SspDE-1d
88									*64.45*	EPSDE-30
89									*65.43*	AdapSS-JADE-10d
90									*66.48*	SADE-20
91									*66.61*	RB-SADE-20
92									*67.68*	MDE_pBX-1d
93									*67.95*	EPSDE-1d
94									*68.75*	CoBiDE-5d
95									*68.75*	SspDE-20
96									*70.00*	CDE-200
97									*70.20*	FDSADE-B
98									*74.52*	EPSDE-20
99									*77.68*	CDE-1d
100									*78.68*	CDE-10d
101									*78.80*	CoBiDE-200
102									*79.20*	CoBiDE-10d
103									*79.57*	CDE-5d
104									*80.11*	CDE-20
105									*81.14*	MDE-pBX-20
106									*81.34*	CDE-3d
107									*82.18*	CDE-2d

of real-world problems the best results are obtained either with algorithms with fixed and dimensionality independent population size (CoBiDE with low *PS*, or Rcr-JADE and JADE-EIG with high *PS*), or ATPS-JADE that use population size adaptation scheme.

6. Apart from ATPS-JADE, DE variants with flexible population size do not show outstanding performance when compared to DE variants with properly selected but fixed population size. On the other hand, with the exception of FDSADE-based approaches, they are never ranked among 20 worst variants. It seems that it is safer to use even not the best DE algorithm with variable population size than the very good DE method with un-properly chosen population size (see the discussion in point 4). One may expect that implementing some population size adaptation rules into the best DE algorithms that are currently used with fixed population size may lead to noticeably improvement.

Some observations regarding the performance of specific algorithms and their population size settings are also worth a few words:

1. Rcr-JADE with *PS* set to 100 or 150 individuals is ranked among 4 best variants in each comparison. The variant of Rcr-JADE with $PS = 100$ is always among 5 best methods, what makes it the most robust among all tested ones.
2. CoBiDE with $PS = 50$ is ranked the third best variant for 2-dimensional CEC2005 benchmarks, and the best one for CEC2011 and 10-dimensional CEC2005 problems. It is, however, only 10th and 21st best variant for 30- and 50-dimensional CEC2005 problems, respectively.
3. ATPS-JADE performance is not outstanding for low-dimensional CEC2005 problems, and this variant is ranked 9th for 30-dimensional CEC2005 problems. However, ATPS-JADE turns out

the best method for 50-dimensional CEC2005 benchmarks and 2nd best for CEC2011 problems.

4. JADE-EIG with $PS=100$ is among the best 11 variants in each considered comparison.
5. CDE with some population size setting (the lowest considered – for low-dimensional CEC2005 problems, 10*d* – for higher-dimensional CEC2005 benchmarks and 2*d*–3*d* – for real-world problems) is always ranked among two the poorest variants, showing large sensitivity to the population size.

5.4. Relation between population size and number of function calls

All the above discussion is based on the final results, obtained when the maximum number of function calls is exploited. As such maximum numbers of function calls are set the same as suggested in CEC2005 and CEC2011 reports, and were applied in plenty of comparisons among Evolutionary Algorithms, using them seems justified. However, various practical problems may require much lower number of function calls, and one may expect that in such a case the results may differ. To verify that, in this section the results obtained during runs after pre-specified number of function calls are analyzed for two sets of problems – real-world CEC2011 and 50-dimensional variants of CEC2005 artificial benchmarks. In case of CEC2011, for which 150,000 calls were allowed, during-run results obtained after 10,000 and 50,000 function calls are chosen; in case of 50-dimensional CEC2005 problems, for which the maximum number of function calls was as high as 500,000, the results obtained after 10,000 and 100,000 function calls are considered. Such two numbers of function calls are selected to show the ranking of population sizes, and ranking of DE variants, after just the initial part of the run (i.e. 10,000 function calls) and when the searching process is developed (i.e. after 50,000 or 100,000 function calls). The similar procedure of comparison is applied as for the final results (i.e. those obtained after maximum number of function calls – see Sections 5.1–5.3). Rankings of various population sizes for each among 10 considered algorithm with fixed population size are given in Table 7 (for results obtained after 10,000 calls) and 8 (for those obtained after 50,000 or 100,000 calls). Tables 9 and 10 show the performance of DE algorithms with flexible population size, and Tables 11 and 12 present the aggregated final rankings of all considered DE variants, with indicated statistical significance of the differences between the particular algorithm and the best one.

It must be noted here that, however, for some algorithms such a procedure may be slightly unfair, namely those that relate the control parameters to the number of function calls still allowed for search. For example jDElscop manage its population size in such a way to perform exploration during earlier parts of the run, and concentrate on exploitation in the late part. Hence, when the performance is tested in the early stages, jDElscop did not start to converge yet. This is probably the case why jDElscop performed so poorly in some tests based on the convergence speed [34].

When evaluating population sizes according to the results obtained at the early stage of the search (10,000 function calls, see Table 7) the variants of each algorithm with the lowest *PS* – i.e. those that allow quickest convergence – turn out the best for CEC2005 artificial benchmarks, and the differences are almost always statistically significant. In case of real-world problems the picture is not as clear – the lowest tested population size ($PS = 20$) is the best only for two algorithms out of ten, others perform better with $PS = 30$, and the differences between results obtained with *PS* set to 20, 30 or 50 are almost never statistically significant. One may conclude that if one has very little time, the low population sizes are advised. However, at least in case of real-world problems – this does not mean as low *PS* as possible.

Results obtained after 50,000 (for CEC2011) or 100,000 (for CEC2005) problems show (see Table 8) that higher population sizes would be recommended than those suggested with respect to the results obtained after 10,000 calls. For 50-dimensional CEC2005 benchmarks after 100,000 function calls the best performances are obtained by variants tested with $PS=100$ (what means 2*d*) or 3*d*. Like in comparisons based on the results obtained after maximum number of function calls, two algorithms require different population size settings – CoBiDE prefers 1*d* and MDE_pBX – 5*d* (see Table 8). For real-world problems variants with population size fixed to 50 or (just in 2 cases) 100 individuals perform best after 50,000 function calls.

Generally speaking, the above discussion confirms what could be expected – that the less time one has to solve the problem, the lower population size is advised. However, as one may note from Table 7, at least in case of real-world problems the population size should not be too low. The more time one has – the higher population size leads to better results (compare Tables 1, 7 and 8).

Comparison of DE variants with flexible *PS* (Tables 9 and 10) may show interesting properties of some algorithms. After 10,000 function calls FDSADE turns out the absolute winner both for 50-

Table 7
Averaged ranking of population sizes of each algorithm based on the results obtained after first 10,000 function calls. The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	50-Dimensional CEC2005 problems											
1*d*=50	**1.12**	**1.28**	**1.52**	**1.24**	**1.08**	**1.32**	**1.20**	**1.36**	**1.40**	**1.08**	**1.26**	**10**
2*d*=100	*1.96*	*2.20*	1.96	1.96	*2.00*	2.00	*2.16*	1.80	1.96	*2.12*	2.01	**0**
3*d*	*3.08*	*2.96*	*2.80*	*3.00*	*3.04*	*2.96*	*2.84*	*3.08*	*2.88*	*3.08*	2.97	**0**
5*d*	*3.92*	*3.76*	*3.80*	*3.96*	*4.08*	*3.84*	*4.00*	*3.92*	*3.88*	*3.88*	3.90	**0**
10*d*	*4.92*	*4.80*	*4.92*	*4.84*	*4.80*	*4.88*	*4.80*	*4.84*	*4.88*	*4.84*	4.85	**0**
	CEC2011 problems											
1*d*	*5.07*	4.39	*5.52*	4.57	4.61	*5.11*	*4.66*	*5.02*	*4.80*	*4.14*	4.79	**0**
2*d*	*5.52*	*5.23*	*5.89*	*5.41*	*5.27*	*5.73*	*5.66*	*5.86*	*5.59*	*5.32*	5.55	**0**
3*d*	*5.98*	*6.05*	*6.52*	*6.05*	*6.09*	*5.95*	*6.07*	*6.09*	*6.23*	*6.27*	6.13	**0**
5*d*	*7.39*	*7.52*	*7.36*	*7.50*	*6.98*	*6.91*	*7.48*	*7.32*	*7.32*	*7.68*	7.35	**0**
10*d*	*8.61*	*8.86*	*8.09*	*8.50*	*8.64*	*8.34*	*8.66*	*8.27*	*8.59*	*8.77*	8.53	**0**
20	2.89	2.86	*4.20*	3.32	3.50	3.84	2.70	**2.59**	3.00	**1.68**	3.06	**2**
30	**2.61**	**2.48**	**2.41**	**2.95**	**2.98**	**3.05**	**2.39**	2.86	**2.68**	2.82	**2.72**	**8**
50	3.57	3.86	3.23	3.00	3.27	3.34	3.93	3.32	3.09	*4.27*	3.49	**0**
100	*5.61*	*5.95*	*5.09*	*5.82*	*5.73*	4.98	*5.93*	*5.91*	*5.86*	*6.11*	5.70	**0**
200	*7.75*	*7.80*	*6.68*	*7.89*	*7.93*	*7.75*	*7.52*	*7.75*	*7.84*	*7.93*	7.68	**0**

Table 8
Averaged ranking of population sizes of each algorithm based on the results obtained after first 50,000 (in case of real-world problems from CEC2011) or 100,000 (in case of CEC2005 benchmark set) function calls. The ranking of particular population size is given in *italic* if the difference between the results achieved with this population size and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Population size	CDE	EPSDE	MDE_pBX	SspDe	SADE	RB-SADE	AdapSS-JADE	Rcr-JADE	JADE-EIG	CoBiDE	**Mean rank**	**Best choice**
	50-Dimensional CEC2005 problems											
1*d*=50	1.66	2.40	3.20	2.84	2.64	3.00	2.82	3.26	*3.42*	**1.08**	2.63	**1**
2*d*=100	**1.62**	**2.24**	2.88	**2.64**	**2.28**	**2.34**	2.66	2.64	2.70	*2.04*	**2.40**	**5**
3*d*	*2.84*	2.56	2.68	2.84	2.72	2.76	**2.40**	**2.44**	**2.46**	*3.04*	2.67	**3**
5*d*	*3.88*	*3.28*	**2.64**	2.96	3.12	3.10	2.68	2.68	2.50	*3.96*	3.08	**1**
10*d*	*5.00*	*4.52*	3.60	*3.72*	*4.24*	*3.80*	*4.44*	*3.98*	*3.92*	*4.88*	4.21	**0**
	CEC2011 problems											
1*d*	*5.34*	4.98	*6.52*	5.25	5.27	5.84	*5.20*	5.25	5.02	*5.05*	5.37	**0**
2*d*	*6.20*	*5.52*	*5.98*	5.34	5.36	5.82	*6.11*	*5.75*	5.66	*5.68*	5.74	**0**
3*d*	*6.48*	*5.89*	*5.80*	6.30	6.23	5.52	*6.59*	*6.39*	*5.89*	*6.23*	6.13	**0**
5*d*	*7.07*	*6.59*	*5.91*	*6.89*	*6.43*	*6.61*	*6.82*	*6.98*	*6.75*	*7.23*	6.73	**0**
10*d*	*7.70*	*8.43*	*7.45*	*7.25*	*7.64*	*6.91*	*7.50*	*7.30*	*8.02*	*8.32*	7.65	**0**
20	4.70	5.11	*7.34*	5.52	5.77	6.14	*5.14*	*5.61*	5.66	*3.91*	5.49	**0**
30	3.52	4.27	4.91	4.89	4.61	4.93	3.77	4.98	4.34	2.59	4.28	**0**
50	**3.02**	**3.30**	4.32	**4.25**	**4.09**	**4.05**	**2.82**	4.11	**3.61**	**2.91**	**3.65**	**8**
100	4.34	4.20	**3.00**	4.34	4.27	4.41	4.18	**3.39**	4.07	*5.59*	4.18	**2**
200	*6.61*	*6.70*	3.77	4.98	5.32	4.77	*6.86*	5.25	*5.98*	*7.50*	5.78	**0**

Table 9
Ranking of algorithms with variable population size after 10,000 function calls. The ranking of particular algorithm is given in *italic* if the difference between the results achieved with this method and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Algorithm	50-Dimensional CEC2005	CEC2011	**Mean rank**	**Best choice**
jDElscop	*6.68*	*6.27*	6.48	**0**
FDSADE	**1.20**	**2.18**	**1.69**	**2**
FDSADE-A	*4.88*	*4.55*	4.71	**0**
FDSADE-B	*5.88*	*5.55*	5.71	**0**
ATPS-JADE	2.20	3.14	2.67	**0**
SapsDE	*3.60*	3.45	3.53	**0**
SapsDE-A	*3.56*	2.86	3.21	**0**

Table 10
Ranking of algorithms with variable population size based on the results obtained after first 50,000 (in case of real-world problems from CEC2011) or 100,000 (in case of CEC2005 benchmarks) function calls. The ranking of particular algorithm is given in *italic* if the difference between the results achieved with this method and the best one (**bolded**) is statistically significant at 5% significance level according to Holm's procedure.

Algorithm	50-Dimensional CEC2005	CEC2011	**Mean rank**	**Best choice**
jDElscop	*6.56*	*5.41*	5.98	**0**
FDSADE	3.16	3.09	3.13	**0**
FDSADE-A	*4.72*	*5.00*	4.86	**0**
FDSADE-B	*6.16*	*5.86*	6.01	**0**
ATPS-JADE	**2.00**	3.23	2.61	**1**
SapsDE	3.12	3.27	3.20	**0**
SapsDE-A	2.28	**2.14**	**2.21**	**1**

dimensional benchmarks from CEC2005 set and for CEC2011 real-world problems (see Table 9), although it is ranked only the second poorest in both such tests when maximum number of function calls specified in [30,130] is used (see Table 5). But after 50,000 (for CEC2011) or 100,000 (for CEC2005) calls, FDSADE is in the middle of the flock, and ATPS-DE (in case of 50-dimensional CEC2005) and SapsDE-A (in case of CEC2011) perform the best (see Table 10). As both ATPS-DE and SapsDE-A also are ranked the first and the second best approach, respectively, after maximum number of function calls is reached, the success of FDSADE achieved after just 10,000 calls shows rather its premature convergence.

When all tested DE variants are compared together after the very early stages of the run (10,000 calls, see Table 11), one obtains fully different ranking to the one noted when the maximum allowed number of function calls is used (Table 6). ATPS-DE, Rcr-JADE-5*d* and Rcr-JADE-3*d*, the first three variants in 50-dimensional CEC2005 competition after maximum number of function calls, are ranked only 25th, 28th and 18th (out of 57 variants) after 10,000 function calls. CoBiDE-50, ATPS-DE and Rcr-JADE-200, the first three variants on CEC2011 real-world problems after maximum number of function calls, are ranked 50th, 55th and 84th (out of 107 variants) after 10,000 calls. On the other hand, variants that win after 10,000 calls (AdapSS-JADE-1*d*, Rcr-JADE-1*d* and JADE-EIG-1*d* in case of 50-dimensional CEC2005 and JADE-EIG-30, Rcr-JADE-30 and Rcr-JADE-20 in case of CEC2011) perform poorly when the maximum number of function calls is exploited.

After 50,000 (for CEC2011) or 100,000 (for CEC2005) function calls, when all DE variants tested are compared together, ranking is again shuffled (see Table 12). The best performing variants are Rcr-JADE-2*d*, AdapSS-JADE-3*d* and ATPS-DE (the winner after maximum number of function calls) in case of CEC2005 benchmarks and CoBiDE-30, Rcr-JADE-100 and JADE-EIG-50 in case of real-world CEC2011 problems. One may note that the population sizes of the winners are generally higher than those of the winners in competition based on the results achieved after 10,000 function calls, but lower than the population sizes of the winners chosen when the maximum number of function calls is used.

Some algorithm-specific conclusions may be also of interest. ATPS-DE is the algorithm with flexible population size that ranked 1st or 2nd on 50-dimensional CEC2005 and CEC2011 problems after maximum number of function calls, but perform poorly on such problems after initial 10,000 calls. Interestingly, after 100,000 calls in case of CEC2005 benchmarks ATPS-DE appears as the 3rd best approach, hence almost as well as at when the maximum number of function calls is used. However, on CEC2011 problems after 50,000 calls ATPS-DE remain only 44th best method, not much better than after 10,000 calls. This may indicate that for various problems population-size adaptive ATPS-DE require roughly 50,000–100,000 function calls to develop its advantage over other competitors.

One should also put attention to the fact that in all rankings, performed after 10,000, 50,000, 100,000, or after maximum number of function calls, Rcr-JADE and JADE-EIG are at the top of the lists, but each time when they are applied with much different

Table 11
Ranking of all considered DE variants (here variant is understood as the algorithm with the specified population size; e.g. if one algorithm is tested with three population sizes, there are three variants) based on the results obtained after first 10,000 function calls. Differences between DE variants which "mean ranks" are given in *italic* and the best variant (ranked 1) are statistically significant at 5% significance level according to Holm's procedure.

	50-Dimensional CEC2005		CEC2011	
Ranking	Mean ranks	DE variant	Mean ranks	DE variant
1	4.48	AdapSS-JADE-1*d*	16.75	JADE-EIG-30
2	4.58	Rcr-JADE-1*d*	17.07	Rcr-JADE-30
3	7.28	JADE-EIG-1*d*	18.11	Rcr-JADE-20
4	7.84	Rcr-JADE-2*d*	19.70	JADE-EIG-20
5	8.22	RB-SADE-1*d*	20.07	Rcr-JADE-50
6	9.96	MDE_pBX-1*d*	20.48	JADE-EIG-50
7	11.00	JADE-EIG-2*d*	21.66	MDE-pBX-30
8	12.30	SADE-1*d*	22.77	RB-SADE-30
9	12.40	MDE_pBX-2*d*	22.89	CoBiDE-20
10	13.66	RB-SADE-2*d*	24.11	AdapSS-JADE-30
11	13.84	EPSDE-1*d*	24.25	MDE_pBX-50
12	13.98	AdapSS-JADE-2*d*	25.98	RB-SADE-50
13	15.64	SspDE-1*d*	27.14	SADE-30
14	15.68	JADE-EIG-3*d*	28.07	AdapSS-JADE-20
15	16.88	MDE_pBX-3*d*	28.61	RB-SADE-20
16	*17.96*	CDE-1*d*	29.30	SspDE-30
17	*18.00*	FDSADE	30.34	EPSDE-30
18	*18.80*	Rcr-JADE-3*d*	30.52	SspDE-50
19	*20.10*	AdapSS-JADE-3*d*	31.80	SADE-50
20	*20.76*	SADE-2*d*	31.84	SspDE-20
21	*20.84*	RB-SADE-3*d*	32.07	SADE-20
22	*22.68*	SspDE-2*d*	33.11	EPSDE-20
23	*23.04*	MDE_pBX-5*d*	35.02	CDE-30
24	*23.36*	EPSDE-2*d*	35.43	AdapSS-JADE-50
25	*23.80*	ATPS-DE	35.59	JADE-EIG-1*d*
26	*25.64*	JADE-EIG-5*d*	36.98	CoBiDE-30
27	*28.04*	CoBiDE-1*d*	37.18	MDE-pBX-20
28	*28.58*	Rcr-JADE-5*d*	39.32	CDE-20
29	*29.40*	SapsDE-A	40.25	EPSDE-50
30	*30.12*	SADE-3*d*	40.36	Rcr-JADE-1*d*
31	*30.28*	CDE-2*d*	40.93	JADE-EIG-2*d*
32	*30.72*	EPSDE-3*d*	41.48	RB-SADE-100
33	*31.00*	SapsDE	42.23	RB-SADE-1*d*
34	*32.04*	SspDE-3*d*	43.93	FDSADE
35	*32.46*	AdapSS-JADE-5*d*	44.20	CDE-50
36	*33.12*	RB-SADE-5*d*	*44.82*	AdapSS-JADE-1*d*
37	*36.96*	MDE_pBX-10*d*	*45.52*	MDE_pBX-100
38	*37.92*	EPSDE-5*d*	*46.20*	Rcr-JADE-2*d*
39	*39.28*	CDE-3*d*	*46.32*	SspDE-1*d*
40	*39.36*	FDSADE-A	*46.59*	SADE-1*d*
41	*39.56*	JADE-EIG-10*d*	*46.68*	MDE_pBX-1*d*
42	*41.12*	CoBiDE-2*d*	*46.98*	JADE-EIG-100
43	*41.54*	SADE-5*d*	*49.59*	EPSDE-1*d*
44	*41.82*	Rcr-JADE-10*d*	*49.66*	JADE-EIG-3*d*
45	*42.36*	SspDE-5*d*	*49.73*	MDE_pBX-2*d*
46	*42.80*	AdapSS-JADE-10*d*	*50.89*	Rcr-JADE-100
47	*45.64*	CDE-5*d*	*51.39*	Rcr-JADE-3*d*
48	*46.28*	CoBiDE-3*d*	*52.30*	SapsDE-A
49	*46.30*	RB-SADE-10*d*	*52.73*	RB-SADE-2*d*
50	*47.08*	EPSDE-10*d*	*53.93*	CoBiDE-50
51	*48.32*	SADE-10*d*	*54.05*	MDE_pBX-3*d*
52	*49.08*	CoBiDE-5*d*	*54.55*	AdapSS-JADE-2*d*
53	*50.02*	SspDE-10*d*	*54.68*	RB-SADE-3*d*
54	*51.16*	FDSADE-B	*54.93*	SADE-2*d*
55	*52.32*	CDE-10*d*	*56.02*	ATPS-DE
56	*52.74*	CoBiDE-10*d*	*56.20*	EPSDE-2*d*
57	*52.86*	jDElscop	*56.30*	SspDE-100
58			*56.61*	SapsDE
59			*56.66*	SspDE-2*d*
60			*57.20*	SADE-100
61			*57.30*	CoBiDE-1*d*
62			*57.34*	AdapSS-JADE-100
63			*58.43*	MDE-pBX-200
64			*60.34*	AdapSS-JADE-3*d*
65			*61.43*	MDE_pBX-5*d*
66			*62.25*	SspDE-3*d*
67			*62.30*	EPSDE-100
68			*62.59*	SADE-3*d*
69			*62.61*	JADE-EIG-5*d*
70			*62.84*	CDE-100
71			*63.07*	EPSDE-3*d*
72			*63.32*	RB-SADE-5*d*
73			*64.39*	Rcr-JADE-5*d*
74			*64.82*	CDE-1*d*
75			*65.52*	CoBiDE-2*d*
76			*66.11*	CDE-3*d*
77			*66.34*	MDE_pBX-10*d*
78			*68.00*	CDE-2*d*
79			*68.27*	JADE-EIG-200
80			*69.20*	FDSADE-A
81			*69.57*	CoBiDE-3*d*
82			*70.27*	SADE-5*d*
83			*71.18*	RB-SADE-200
84			*71.18*	Rcr-JADE-200
85			*72.27*	AdapSS-JADE-200
86			*72.82*	AdapSS-JADE-5*d*
87			*73.16*	SspDE-5*d*
88			*74.16*	CDE-5*d*
89			*75.39*	Rcr-JADE-10*d*
90			*76.00*	EPSDE-5*d*
91			*77.05*	CoBiDE-100
92			*77.16*	JADE-EIG-10*d*
93			*79.25*	RB-SADE-10*d*
94			*79.86*	EPSDE-200
95			*80.75*	AdapSS-JADE-10*d*
96			*81.45*	SspDE-200
97			*81.48*	CoBiDE-5*d*
98			*81.68*	SADE-200
99			*82.70*	CDE-200
100			*84.07*	FDSADE-B
101			*85.07*	SADE-10*d*
102			*86.02*	SspDE-10*d*
103			*86.61*	EPSDE-10*d*
104			*88.52*	CDE-10*d*
105			*90.61*	jDElscop
106			*91.27*	CoBiDE-200
107			91.89	CoBiDE-10*d*

population sizes. This suggests that for various numbers of function calls the same good algorithm may be used, but with much different population size.

5.5. *From another perspective – how often specific Differential Evolution variants perform best*

The discussion given in Sections 5.1–5.4 has been based on the average rankings of DE variants. However, many practitioners may be interested in determining DE algorithm, together with its population size, that performs "best" for many problems. Of course, for over a decade it is widely accepted that no heuristic may outperform others on all possible problems. According to No Free Lunch theorems for optimization [79,161,162], it is known that the performance of any heuristic, including purely random search, averaged over all problems that are closed under permutation [126] will be equal. However, it is also known that the majority of such "all problems" would not be of interest to anyone [73,74], and that for the problems that may be of interest to someone, some heuristics may perform better than the others (but consult [160]).

One may ask if any DE algorithm with particular population size setting perform better than all the others for a number of problems. We restrict our discussion in this section to the results obtained when the maximum number of function calls is used. Table 13 recall the variants that won with all other (56, 76 or 106) DE variants on at least one problem, and count number of their

Table 12
Ranking of all considered DE variants (here variant is understood as the algorithm with the specified population size; e.g. if one algorithm is tested with three population sizes, there are three variants) based on the results obtained after first 50,000 (in case of real-world problems from CEC2011) or 100,000 (in case of CEC2005 benchmarks) function calls. Differences between DE variants which "mean ranks" are given in *italic* and the best variant (ranked 1) are statistically significant at 5% significance level according to Holm's procedure.

	50-Dimensional CEC2005		CEC2011	
Ranking	Mean ranks	DE variant	Mean ranks	DE variant
1	13.08	Rcr-JADE-2*d*	24.16	CoBiDE-30
2	13.28	AdapSS-JADE-3*d*	24.98	Rcr-JADE-100
3	13.40	ATPS-DE	26.98	JADE-EIG-50
4	13.60	Rcr-JADE-3*d*	28.89	Rcr-JADE-50
5	14.44	JADE-EIG-3*d*	29.07	MDE_pBX-100
6	14.94	JADE-EIG-2*d*	29.98	JADE-EIG-100
7	15.00	AdapSS-JADE-2*d*	30.93	MDE-pBX-200
8	15.40	SapsDE-A	33.43	MDE_pBX-50
9	15.90	Rcr-JADE-5*d*	33.61	RB-SADE-50
10	17.24	JADE-EIG-5*d*	33.66	RB-SADE-100
11	17.66	AdapSS-JADE-5*d*	34.39	CoBiDE-50
12	18.50	AdapSS-JADE-1*d*	34.98	CoBiDE-20
13	19.52	RB-SADE-2*d*	35.02	AdapSS-JADE-50
14	19.78	SapsDE	35.23	JADE-EIG-30
15	20.36	Rcr-JADE-1*d*	36.30	SapsDE-A
16	22.14	RB-SADE-3*d*	36.84	SADE-50
17	23.14	CoBiDE-1*d*	37.61	Rcr-JADE-30
18	23.26	FDSADE	37.75	SADE-100
19	23.82	JADE-EIG-1*d*	40.73	RB-SADE-30
20	24.56	RB-SADE-1*d*	41.16	Rcr-JADE-200
21	24.78	MDE_pBX-2*d*	41.36	JADE-EIG-1*d*
22	24.82	MDE_pBX-3*d*	41.98	EPSDE-50
23	24.88	MDE_pBX-5*d*	42.34	AdapSS-JADE-30
24	25.16	RB-SADE-5*d*	42.82	Rcr-JADE-1*d*
25	25.74	MDE_pBX-1*d*	42.89	RB-SADE-200
26	26.30	Rcr-JADE-10*d*	43.07	JADE-EIG-2*d*
27	26.40	SADE-2*d*	43.20	SspDE-100
28	*26.90*	SADE-1*d*	43.39	SspDE-50
29	*27.12*	CDE-1*d*	43.52	JADE-EIG-20
30	*28.02*	SADE-3*d*	43.70	MDE-pBX-30
31	*28.16*	JADE-EIG-10*d*	43.73	SADE-30
32	*30.22*	CDE-2*d*	43.84	Rcr-JADE-2*d*
33	*30.46*	SspDE-5*d*	44.25	AdapSS-JADE-100
34	*30.64*	SADE-5*d*	44.43	Rcr-JADE-20
35	*31.06*	EPSDE-3*d*	46.93	CDE-50
36	*31.20*	SspDE-2*d*	46.93	EPSDE-100
37	*31.80*	MDE_pBX-10*d*	47.27	RB-SADE-3*d*
38	*31.82*	SspDE-3*d*	47.43	JADE-EIG-200
39	*31.90*	RB-SADE-10*d*	47.98	JADE-EIG-3*d*
40	*32.42*	SspDE-1*d*	49.16	FDSADE
41	*32.44*	EPSDE-2*d*	49.20	AdapSS-JADE-20
42	*33.50*	AdapSS-JADE-10*d*	49.36	EPSDE-30
43	*33.78*	EPSDE-1*d*	49.48	ATPS-DE
44	*35.04*	EPSDE-5*d*	49.70	SspDE-30
45	*36.40*	CDE-3*d*	49.77	SADE-1*d*
46	*36.68*	SspDE-10*d*	50.11	SADE-2*d*
47	*39.68*	SADE-10*d*	50.18	MDE_pBX-2*d*
48	*39.96*	FDSADE-A	50.30	RB-SADE-2*d*
49	*40.88*	CoBiDE-2*d*	50.48	SapsDE
50	*43.96*	CDE-5*d*	50.73	RB-SADE-1*d*
51	*44.98*	EPSDE-10*d*	51.59	AdapSS-JADE-1*d*
52	*46.30*	CoBiDE-3*d*	51.66	SADE-200
53	*50.38*	CoBiDE-5*d*	51.70	Rcr-JADE-3*d*
54	*51.88*	CDE-10*d*	51.93	SspDE-200
55	*52.46*	FDSADE-B	52.23	MDE_pBX-3*d*
56	*52.58*	jDElscop	*53.48*	SspDE-2*d*
57	*53.28*	CoBiDE-10*d*	*53.57*	CDE-30
58			*54.48*	RB-SADE-20
59			*54.59*	SspDE-1*d*
60			*54.66*	MDE_pBX-5*d*
61			*55.39*	JADE-EIG-5*d*
62			*55.66*	SADE-20
63			*56.05*	MDE_pBX-1*d*
64			*56.14*	RB-SADE-5*d*
65			*56.39*	Rcr-JADE-5*d*
66			*56.68*	SADE-3*d*
67			*56.98*	SspDE-20
68			*58.25*	CDE-100
69			*58.43*	SspDE-3*d*
70			*59.05*	EPSDE-1*d*
71			*59.39*	EPSDE-20
72			*60.48*	CoBiDE-1*d*
73			*60.98*	RB-SADE-10*d*
74			*62.52*	SADE-5d
75			*62.91*	AdapSS-JADE-2*d*
76			*63.18*	CDE-20
77			*63.30*	EPSDE-2*d*
78			*63.39*	MDE_pBX-10*d*
79			*64.20*	JADE-EIG-10*d*
80			*64.43*	AdapSS-JADE-200
81			*64.82*	MDE-pBX-20
82			*64.93*	AdapSS-JADE-3*d*
83			*65.43*	Rcr-JADE-10*d*
84			*65.89*	SspDE-5*d*
85			*66.66*	EPSDE-3*d*
86			*68.18*	AdapSS-JADE-5*d*
87			*68.20*	CoBiDE-2*d*
88			*68.43*	EPSDE-200
89			*69.98*	FDSADE-A
90			*70.82*	EPSDE-5*d*
91			*70.93*	CoBiDE-3*d*
92			*71.07*	SspDE-10*d*
93			*71.11*	SADE-10*d*
94			*71.16*	CoBiDE-100
95			*72.36*	CDE-1*d*
96			*73.16*	AdapSS-JADE-10*d*
97			*76.43*	CoBiDE-5*d*
98			*78.84*	CDE-3*d*
99			*79.09*	CDE-5*d*
100			*79.34*	CDE-200
101			*80.73*	CDE-2*d*
102			*82.07*	EPSDE-10*d*
103			*82.11*	FDSADE-B
104			*82.11*	jDElscop
105			*83.66*	CDE-10*d*
106			*85.70*	CoBiDE-200
107			*85.89*	CoBiDE-10*d*

wins (the number of such "wins" is often lower than the number of problems, due to ties (consult section 4.1) that are not counted).

The superiority of some variants over all the others was noted just on 3 out of 25 2-dimensional CEC2005 problems. This confirms that 2-dimensional artificial benchmarks are not very selective. No algorithm gained more than one win. When dimensionality of the CEC2005 problems increases, more frequently the best variant for particular problem may be determined. However, the most selective are CEC2011 real-world problems – in 17 out of 22 such problems one variant turned out better than all the others.

It may be, however, very surprising that, putting aside 10-dimensional CEC2005 problems, many variants occasionally wins for one or two problems, but none gain more than 3 wins per comparison (22 or 25 problems). DE algorithms with contradictory population size settings may win for different specific problems. This is especially noticeable for CEC2011 ones – among variants that performed best for specific real-world problems some uses *PS* set to 1*d*, 2*d* or 20, others – to 10*d* or 200. But the most surprising is that in case of real-world problems the three variants that are considered best according to the average ranks (CoBiDE-50, ATPS-JADE and Rcr-JADE-200, consult Table 6) do not gained… any win. It means that, although they "on average" perform well, they would not be especially recommended for any particular application.

DE algorithms with flexible population size very rarely perform best for the particular problem. ATPS-JADE, which is ranked

Table 13
A summary of DE variants that outperform all other variants (see section 4.1 for the definition of ties) on at least one problem when the maximum allowed number of function calls is used.

2-Dimensional CEC2005		10-Dimensional CEC2005		30-Dimensional CEC2005		50-Dimensional CEC2005		CEC2011	
Nr of wins	DE variant	Nr of wins	DE variant	Nr of wins	DE variant	Nr of wins	DE variant	Nr of wins	DE variant
1	EPSDE-100	8	CDE-3*d*	2	CoBiDE-50	3	JADE-EIG-10*d*	2	JADE-EIG-3*d*
1	Rcr-JADE-100	4	CoBiDE-10*d*=100	2	SADE-100	2	EPSDE-5*d*	2	CDE-50
1	SspDE-100	1	JADE-EIG-10*d*=100	1	Rcr-JADE-2*d*	2	RB-SADE-3*d*	1	JADE-EIG-10*d*
		1	Rcr-JADE-5*d*=50	1	Rcr-JADE-3*d*	1	AdapSS-JADE-1*d*=50	1	RB-SADE-20
		1	Rcr-JADE-10*d*=100	1	JADE-EIG-10*d*	1	AdapSS-JADE-3*d*	1	RB-SADE-200
		1	RB-SADE-10*d*=100	1	AdapSS-JADE-1*d*	1	AdapSS-JADE-5*d*	1	RB-SADE-10*d*
		1	jDElscop	1	SADE-50	1	RcrJADE-2*d*=100	1	SADE-200
				1	RB-SADE-1*d*	1	Rcr-JADE-5*d*	1	Rcr-JADE-50
				1	SspDE-3*d*	1	CoBiDE-2*d*=100	1	Rcr-JADE-1*d*
				1	SspDE-100	1	EPSDE-3*d*	1	Rcr-JADE-10*d*
				1	CoBiDE-3*d*	1	MDE_pBX-1*d*=50	1	Rcr-JADE-3*d*
				1	MDE_pBX-50	1	SapsDE	1	AdapSS-JADE-1*d*
						1	ATPS-JADE	1	CoBiDE-3*d*
								1	CoBiDE-2*d*
								1	jDElscop

among two best variants on 50-dimensional CEC2005 benchmarks and CEC2011 problems, wins with all competitive variants only once (on 50-dimensional CEC2005 problem F14); jDElscop wins once for real-world CEC2011 and once for 10-dimensional CEC2005 problems. This suggests that the flexible population size helps avoiding pitfalls, but is not enough to elaborate the outstanding performance.

Only one DE variant gains large number of wins in one comparison – CDE-3*d* outperforms all other variants on 8 out of 25 10-dimensional CEC2005 problems. With surprise CDE-3*d*, according to the averaged ranking, would be considered only 29th best method for 10-dimensional CEC2005 problems (see Table 6). Moreover, CDE with slightly lower population size (CDE-2*d*) is, according to the averaged ranking, among 5 poorest variants on these problems. For all 5 performed comparisons, according to the averaged rankings CDE-3*d* is always within the poorer half of tested variants; in case of 2-dimensional CEC2005 problems it is even the worst, in case of real-world problems – the second worst variant! Hence, the variant that show so bright performance on some among 10-dimensional problems is, based on other criteria, one of the poorest. Detailed look at the results achieved by CDE-3*d* revealed that it indeed perform perfectly well for many, but less than 50% of 10-dimensional CEC2005 problems (see Supplementary Table 2). As its performance is among the poorest for almost all other problems, CDE-3*d* may be considered over-fitted to specific artificial benchmarks (it perform well for just one (F13) real-world problem considered in this study). This brief discussion show how fragile the search for the best DE algorithms and their population size settings is. It also reveals how un-trustworthy the comparisons based on a few selected artificial benchmark problems may be.

Based on the results from Tables 6 and 13 one cannot expect that variants that are on average ranked best would allow finding best solutions of specific problems the particular user is interested in. Rather, DE variants which win competitions are those that avoid pitfalls.

6. Conclusions

The present study discusses the impact of the population size on the performance of various Differential Evolution algorithms and aims to determine some rules how to set the population size, and whether it should be fixed or adaptive during run. The study is based on both numerical benchmark (CEC2005) and real-world problems (CEC2011); conclusions 1-5 are valid when the allowed numbers of function calls defined in the source publications of CEC2005 and CEC2011 [130,30] are used, but convergence speed is also adressed in conclusion 6. Based on the achieved results, the main questions asked in the Introduction may be answered as follows:

1. The inappropriate choice of the population size, even when only values frequently used in the literature are considered, may hamper the performance of each Differential Evolution algorithm. The importance of the population size setting for the performance of the algorithm vary, depending not only on the specific method, but also on the problem to be solved and, in case of scalable problems, dimensionality (see Section 5.3).
2. Setting Differential Evolution population size lower than 50 individuals is rarely recommended even for low dimensional problems. Too low population sizes may diminish the number of available moves and prevent convergence within the specified number of function calls, even in case of unimodal problems (see Supplementary Tables 1 and 2). The impact of very small population size on the performance of Differential Evolution algorithms tested on hybrid composition functions or real-world problems is clearly unfavorable (consult Tables 3 and 4).
3. The population size fixed to 100 individuals, which is frequently used in recent years, is rarely a poor choice for low-dimensional problems ($d<30$, where d is the problem dimensionality). However, for higher-dimensional problems larger population sizes are often needed. Unfortunately, their choice turned out relatively easy only for artificial benchmarks. For such 30- or 50-dimensional artificial problems the population size is recommended to be set to 3*d*–5*d*, the relation similar to the one found in [50] over decade ago, that was based on analysis of just a very few basic non-adaptive Differential Evolution variants tested on extremely simple problems. But for higher-dimensional real-world problems the choice of the proper population size severely depends on the problem and specific algorithm. Using 200 individuals may be an initial guess, but many Differential Evolution algorithms would perform better when population size is set to some value between [50,10*d*].
4. The majority of already proposed Differential Evolution algorithms with flexible population size do not outperform the variants with fixed population size, assuming that this population size has been chosen properly. However, for higher-dimensional artificial CEC2005, or real-world CEC2011 problems ATPS-JADE [175] with adaptive population size turns out the best and the second best among Differential Evolution variants. This is very encouraging, as the user of ATPS-JADE does not

need to pay attention to initial adjustment of the population size. Overall, the population size adaptation is not much needed for low-dimensional problems, but turns out very important for higher-dimensional ones.

5. Among population size adaptation algorithms the ATPS-DE [175] turns out the most effective, but only if the number of function calls used is sufficiently large. It seems that including adaptation methods from ATPS-DE into Differential Evolution algorithms that are until now used with fixed population size may noticeably improve their performance, although this concept has not been tested in this study. Other rules aiming at modification of Differential Evolution population size are less successful, but apart from FDSADE-based ones [136], none variant with adaptive population size was ranked among 20 (out of 57, 77 or 107 tested) the worst Differential Evolution variants in any among 5 comparisons performed in this study. This highlights the importance of the population size flexibility for Differential Evolution algorithms. Population size adaptation methods deserve much more attention in the future studies.
6. All above conclusions have been based on the final results, obtained after maximum number of function calls – specified in the reports that introduce test problems used [30,130] – is exploited. However, in this study also a quite clear relation between the population size and the convergence speed is observed. According to the rankings based on the results obtained during-runs after lower number of function calls, the less function calls is used, DE variants with the lower population sizes perform better. However, at least for real-world problems, using too small population sizes is not advised. Even when the number of function calls is as low as 10,000, often 30–50 individuals are needed to get the best results. When the results obtained after very low number of function calls are compared, among algorithms with population size adaptation FDSADE [136] perform better than the other variants tested, including ATPS-DE [175]. As a result, the choice of the best DE algorithm with flexible population size seems related to the number of function calls that may be used.

It has also been found that even if some Differential Evolution algorithms share large similarities (for example those that are the modified variants of JADE or SADE algorithms), they do not necessarily perform best with similar population size.

Finally, when Differential Evolution algorithms are tested with various population sizes on a number of problems, it turns out that very rarely one particular algorithm with specific population size would outperform all other algorithms with all considered population size settings on more than 1–2 problems, out of 22–25. Interestingly, variants with population size settings that receive highest rankings averaged over all considered problems are those that rarely perform poor, not those that were able to find the best solution for some problems. In other words, winners never fail, but would hardly be recommended for any specific application. This points out that although some general rules regarding the choice of the population size may be given, it does not relieve the practitioners from deliberately choosing not only the algorithm, but also the appropriate population size for particular application. To diminish the burden with population size setting, the more attention to the effective population size adaptation methods is needed. Empirical results presented in this paper indicate that algorithms with adaptive population size, although so few, turns out competitive to the best Differential Evolution variants with fixed, deliberately chosen population size.

Acknowledgments

This work was supported within statutory activities No 3841/E-41/S/2015 of the Ministry of Science and Higher Education of Poland.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.swevo.2016.05.003.

References

[1] M.A. Ahandani, Opposition-based learning in the shuffled bidirectional Differential Evolution algorithm, Swarm Evolut. Comput. 26 (2016) 64–85.
[2] A. Ahrari, M. Shariat-Panahi, An improved evolution strategy with adaptive population size, Optimization 64 (12) (2015) 2567–2586.
[3] M.M. Ali, A. Torn, Population set-based global optimization algorithms: some modifications and numerical studies, Comput. Oper. Res. 31 (10) (2004) 1703–1725.
[4] M.M. Ali, L.P. Fatti, A differential free point generation scheme in the Differential Evolution algorithm, J. Glob. Optim. 35 (4) (2006) 551–572.
[5] M.M. Ali, Differential Evolution with preferential crossover, Eur. J. Oper. Res. 181 (3) (2007) 1137–1147.
[6] M.M. Ali, Differential Evolution with generalized differentials, J. Comput. Appl. Math. 235 (8) (2011) 2205–2216.
[7] M.Z. Ali, N.H. Awad, P.N. Suganthan, Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization, Appl. Soft Comput. 33 (2015) 304–327.
[8] J. Apolloni, J.G. Nieto, E. Alba, G. Leguizamon, Empirical evaluation of distributed Differential Evolution on standard benchmarks, Appl. Math. Comput. 236 (2014) 351–366.
[9] Md Asafuddoula, T. Ray, R. Sarker, An adaptive hybrid Differential Evolution algorithm for single objective optimization, Appl. Math. Comput. 231 (2014) 601–618.
[10] R.L. Becerra, C.A. Coello, Cultured Differential Evolution for constrained optimization, Comput. Methods Appl. Mech. Eng. 195 (33-36) (2006) 4303–4322.
[11] I. Boussaid, A. Chatterjee, P. Siarry, M. Ahmed-Nacer, Two-stage update biogeography-based optimization using differential evolution algorithm (DBBO), Comput. Oper. Res. 38 (8) (2011) 1188–1198.
[12] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in Differential Evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evolut. Comput. 10 (6) (2006) 646–657.
[13] J. Brest, B. Boskovic, S. Greiner, V. Zumer, M.S. Maucec, Performance comparison of self-adaptive and adaptive Differential Evolution algorithms, Soft Comput. 11 (2007) 617–629.
[14] J. Brest, M.S. Maucec, Population size reduction for the Differential Evolution algorithm, Appl. Intell. 29 (3) (2008) 228–247.
[15] J. Brest, M.S. Maucec, Self-adaptive Differential Evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (11) (2011) 2157–2174.
[16] J. Brest, B. Boskovic, A. Zamuda, I. Fister, M.S. Maucec, Self-adaptive Differential Evolution algorithm with a small and varying population size, in: WCCI 2012 IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012.
[17] Z.H. Cai, W.Y. Gong, C.X. Ling, H. Zhang, A clustering-based Differential Evolution for global optimization, Appl. Soft Comput. 11 (1) (2011) 1363–1379.
[18] Y.Q. Cai, J.H. Wang, Differential Evolution with neighborhood and direction information for numerical optimization, IEEE Trans. Cybern. 43 (6) (2013) 2202–2215.
[19] M.P. Camargo, J.L. Rueda, I. Erlich, O. Ano, Comparison of emerging metaheuristic algorithms for optimal hydrothermal system operation, Swarm Evolut. Comput. 18 (2014) 83–96.
[20] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic Differential Evolution frameworks, Soft Comput. 13 (8-9) (2009) 811–831.
[21] F. Caraffini, F. Neri, I. Poikolainen, Micro-differential evolution with extra moves along the axes, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, 2013, pp. 46–53.
[22] T.S. Chen, K. Yang, G.L. Chen, X. Yao, A large population size can be unhelpful in Evolutionary Algorithms, Theor. Comput. Sci. 436 (2012) 54–70.
[23] J.X. Cheng, G.X. Zhang, F. Neri, Enhancing distributed Differential Evolution with multicultural migration for global numerical optimization, Inf. Sci. 247 (2013) 72–93.
[24] M. Clerc, Particle Swarm Optimization, ISTE Ltd, London, UK, 2006.
[25] L.D. Coelho, D.L.D. Bernert, A modified Ant Colony Optimization algorithm based on Differential Evolution for chaotic synchronization, Expert Syst. Appl. 37 (6) (2010) 4198–4203.

[26] M. Crepinsek, S.H. Liu, M. Mernik, Exploration and exploitation in Evolutionary Algorithms: a survey, ACM Comput. Surv. 45 (3) (2013) 1–3335.
[27] M. Crepinsek, S.H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guides how to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[28] R. Das, D.K. Prasad, Prediction of porosity and thermal diffusivity in a porous fin using differential evolution algorithm, Swarm Evolut. Comput. 23 (2015) 27–39.
[29] S. Das, A. Abraham, U.K. Chakraboty, A. Konar, Differential Evolution using a neighborhood-based mutation operator, IEEE Trans. Evolut. Comput. 13 (3) (2009) 526–553.
[30] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Technical Report, Jadavpur Univ., Nanyang Technological Univ, 2010.
[31] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evolut. Comput. 15 (1) (2011) 27–54.
[32] I. De Falco, A. Della Cioppa, D. Maisto, U. Scafuri, E. Tarantino, An adaptive invasion-based model for distributed Differential Evolution, Inf. Sci. 278 (2014) 653–672.
[33] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[34] J. Derrac, S. Garcia, S. Hui, P.N. Suganthan, F. Herrera, Analyzing convergence performance of evolutionary algorithms: a statistical approach, Inf. Sci. 289 (2014) 41–58.
[35] O.J. Dunn, Multiple comparisons among means, J. Am. Stat. Assoc. 56 (293) (1961) 52–64.
[36] M.C. Du Plessis, A.P. Engelbrecht, Using Competitive Population Evaluation in a differential evolution algorithm for dynamic environments, Eur. J. Oper. Res. 218 (1) (2012) 7–20.
[37] A.S. Dymond, A.P. Engelbrecht, S. Kok, P.S. Heyns, Tuning optimization algorithms under multiple objective function evaluation budgets, IEEE Trans. Evolut. Comput. 19 (3) (2015) 341–358.
[38] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th Int. Symp. Micromachine Human Sci., Nagoya, Japan, 1995, pp. 39–43.
[39] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evolut. Comput. 3 (2) (1999) 124–141.
[40] A.E. Eiben, E. Marchiori, V.A. Valko, Evolutionary Algorithms with on-the-fly population size adjustment, in: Proceedings of the 8th Parallel Problem Solving From Nature (PPSN VIII), LNCS 3424, 2004, pp. 41–50.
[41] A.E. Eiben, J. Smith, From evolutionary computation to the evolution of things, Nature 521 (2015) 476–482.
[42] S.M. Elsayed, R.A. Sarker, D.L. Essam, Self-adaptive Differential Evolution incorporating a heuristic mixing of operators, Comput. Optim. Appl. 54 (3) (2013) 771–790.
[43] S.M. Elsayed, R.A. Sarker, Differential Evolution with automatic population injection scheme, in: Proceedings of IEEE Symposium Series on Computational Intelligence, Singapore, 2013, pp.16–19.
[44] S.M. Elsayed, R.A. Sarker, D.L. Essam, A self-adaptive combined strategies algorithm for constrained optimization using Differential Evolution, Appl. Math. Comput. 241 (2014) 267–282.
[45] A.P. Engelbrecht, Fitness function evaluations: a fair stopping condition?, in: Proceedings of IEEE Symposium Series on Computational Intelligence, Orlando, Florida, USA, December 9–12, 2014.
[46] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing Differential Evolution utilizing proximity-based mutation operations, IEEE Trans. Evolut. Comput. 15 (1) (2011) 99–119.
[47] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to Differential Evolution, J. Glob. Optim. 27 (1) (2003) 105–129.
[48] Q.Q. Fan, X.F. Yan, Self-adaptive Differential Evolution algorithm with discrete mutation control parameters, Expert Syst. Appl. 42 (2015) 1551–1572.
[49] V. Feoktisov, Differential Evolution in Search of Solutions, Book Series: Springer Optimization and Its Applications, Springer, New York, 2006.
[50] R. Gamperle, S.D. Muller, P. Koumoutsakos, A parameter study of Differential Evolution, in: Ales Grmela, Nikos E. Mastorakis (Eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS Press, Interlaken, Switzerland, 2002.
[51] Z. Gao, Z. Pan, J. Gao, A new highly efficient Differential Evolution scheme and its application to waveform inversion, IEEE Geosci. Remote Sens. Lett. 11 (10) (2014) 1702–1706.
[52] W.F. Gao, G.G. Yen, S.Y. Liu, A cluster-based Differential Evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (8) (2014) 1314–1327.
[53] W.F. Gao, G.G. Yen, S.Y. Liu, A dual-population Differential Evolution with coevolution for constrained optimization, IEEE Trans. Cybern. 45 (5) (2015) 1094–1107.
[54] S. Garcia, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal. Mach. Learn. Res. 9 (2008) 2677–2694.
[55] S. Garcia, A. Fernandez, J. Luengo, F. Herdera, Advanced nonparametric tests for multiple comparison in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[56] A. Ghosh, S. Das, A. Chowdhury, R. Gini, An improved Differential Evolution algorithm with fitness-based adaptation of the control parameters, Inf. Sci. 181 (18) (2011) 3749–3765.
[57] S. Ghosh, S. Das, S. Roy, S.K.M. Islam, P.N. Suganthan, A Differential Covariance Matrix Adaptation Evolutionary Algorithm for real parameter optimization, Inf. Sci. 182 (2012) 199–219.
[58] M.S. Gibbs, H.R. Maier, G.C. Dandy, Using characteristics of the optimisation problem to determine the Genetic Algorithm population size when the number of evaluations is limited, Environ. Model. Softw. 69 (2015) 226–239.
[59] W.Y. Gong, Z.H. Cai, C.Y. Ling, DE/BBO: a hybrid Differential Evolution with Biogeography-based optimization for global numerical optimization, Soft Comput. 15 (4) (2011) 645–665.
[60] W.Y. Gong, A. Fialho, Z.H. Cai, H. Li, Adaptive strategy selection in Differential Evolution for numerical optimization: an empirical study, Inf. Sci. 181 (24) (2011) 5364–5386.
[61] W.Y. Gong, Z.H. Cai, Differential Evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (6) (2013) 2066–2081.
[62] W.Y. Gong, Z.H. Cai, D.W. Liang, Engineering optimization by means of an improved constrained Differential Evolution, Comput. Methods Appl. Mech. Eng. 268 (2014) 884–904.
[63] W.Y. Gong, Z.H. Cai, Y. Wang, Repairing the crossover rate in adaptive Differential Evolution, Appl. Soft Comput. 15 (2014) 149–168.
[64] W.Y. Gong, Z.H. Cai, D.W. Liang, Adaptive ranking mutation operator based Differential Evolution for constrained optimization, IEEE Trans. Cybern. 45 (4) (2015) 716–727.
[65] V. Gonuguntla, R. Mallipeddi, K.C. Veluvou, Differential Evolution with population and strategy parameter adaptation, Math. Probl. Eng. (2015) 287607.
[66] H.X. Guo, Y.N. Li, J.L. Li, H. Sun, D.Y. Wang, X.H. Chen, Differential evolution improved with self-adaptive control parameters based on simulated annealing, Swarm Evolut. Comput. 19 (2014) 52–67.
[67] S.M. Guo, C.C. Yang, Enhancing Differential Evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evolut. Comput. 19 (1) (2015) 31–49.
[68] S.M. Guo, C.C. Yang, P.H. Hsu, J.S.H. Tsai, Improving Differential Evolution with a successful-parent-selecting framework, IEEE Trans. Evolut. Comput. 19 (5) (2015) 717–730.
[69] N. Hansen, A.S.P. Niederberger, L. Guzzella, P. Koumoutsakos, A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion, IEEE Trans. Evolut. Comput. 13 (1) (2009) 180–197.
[70] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[71] Z.B. Hu, S.W. Xiong, Q.H. Su, X.W. Zhang, Sufficient conditions for global convergence of Differential Evolution algorithm, J. Appl. Math. (2013) 193196.
[72] Z.B. Hu, S.W. Xiong, Q.H. Su, Z.X. Fang, Finite Markov chain analysis of classical Differential Evolution algorithm, J. Comput. Appl. Math. 268 (2014) 121–134.
[73] C. Igel, M. Toussaint, On classes of functions for which no free lunch results hold, Inf. Process. Lett. 86 (6) (2003) 317–321.
[74] C. Igel, M. Toussaint, A no-free lunch theorem for non-uniform distributions of target functions, J. Math. Model. Algorithms 3 (2004) 313–322.
[75] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive Differential Evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybern. Part B – Cybern. 42 (2) (2012) 482–500.
[76] T. Jansen, K.A. De Jong, I. Wegener, On the choice of the offspring population size in Evolutionary Algorithms, Evolut. Comput. 13 (4) (2005) 413–440.
[77] D.L. Jia, G.X. Zheng, M.K. Khan, An effective memetic Differential Evolution algorithm based on chaotic local search, Inf. Sci. 181 (15) (2011) 3175–3187.
[78] P. Kaelo, M.M. Ali, A numerical study on some modified Differential Evolution algorithms, Eur. J. Oper. Res. 169 (3) (2006) 1176–1184.
[79] M. Köppen, D.H. Wolpert, W.G. Macready, Remarks on a recent paper on the "No free lunch" theorems, IEEE Trans. Evolut. Comput. 5 (3) (2001) 295–296.
[80] M. Kotyrba, E. Volna, P. Bujok, Unconventional modelling of complex system via Cellular Automata and Differential Evolution, Swarm Evolut. Comput. 25 (2015) 52–62.
[81] D. Kovacevic, N. Mledenovic, B. Petrovic, P. Milosevic, DE-VNS: Self-adaptive Differential Evolution with crossover neighborhood search for continuous global optimization, Comput. Oper. Res. 52 (2014) 157–169.
[82] S. Kundu, S. Das, A.V. Vasilakos, S. Biswas, A modified differential evolution-based combined routing and sleep scheduling scheme for lifetime maximization of wireless sensor networks, Soft Comput. 19 (3) (2015) 637–659.
[83] J. Lampinen, I. Zelinka, On stagnation of the Differential Evolution algorithm, in: Proceedings of 6th International Mendel Conference on Soft Computing, Brno, Czech Republic, 2000.
[84] G.J. LaPorte, J. Branke, C.H. Chen, Adaptive parent population sizing in Evolution Strategies, Evolut. Comput. 23 (3) (2015) 397–420.
[85] Y.L. Li, Z.H. Zhan, Y.J. Gong, W.N. Chen, J. Zhang, Y. Li, Differential Evolution with an evolution path: A DEEP Evolutionary Algorithm, IEEE Trans. Cybern. 45 (9) (2015) 1798–1810.
[86] C. Lin, A.Y. Qing, Q.Y. Feng, A comparative study of crossover in Differential Evolution, J. Heuristics 17 (6) (2011) 675–703.
[87] J. Liu, J. Lampinen, A fuzzy adaptive Differential Evolution algorithm, Soft Comput. 9 (2005) 448–462.
[88] F.G. Lobo, C.F. Lima, A review of adaptive population sizing schemes in Genetic Algorithms, GECCO (2005) 228–234.
[89] K.M. Malan, A.P. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Inf. Sci. 241 (2013) 148–163.

[90] R. Mallipeddi, P.N. Suganthan, Empirical study on the effect of population size on Differential Evolution algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, Hong Kong, 2008.
[91] R. Mallipeddi, P.N. Suganthan, Differential Evolution algorithm with ensemble of populations for global numerical optimization, OPSEARCH 46 (2) (2009) 184–213.
[92] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential Evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[93] K.L. Mills, J.J. Filliben, A.L. Haines, Determining relative importance and effective settings for Genetic Algorithm control parameters, Evolut. Comput. 23 (2) (2015) 309–342.
[94] A.W. Mohamed, Z.H. Sabry, Constrained optimization based on modified Differential Evolution algorithm, Inf. Sci. 194 (2012) 171–208.
[95] J. Montgomery, Differential Evolution Difference vectors and movement in-solution space, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trond-heim, Norway, 2009.
[96] R. Mukherjee, G.R. Patra, R. Kundu, S. Das, Cluster-based, Differential Evolution with crowding archive for niching in dynamic environments, Inf. Sci. 267 (2014) 58–82.
[97] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evolut. Comput. 2 (2012) 1–14.
[98] F. Neri, V. Tirronen, On memetic Differential Evolution frameworks: a study of advantages and limitations in hybridization, in: Proceedings of the IEEE World Congress on Computational Intelligence, 2008, pp 2135–2142.
[99] F. Neri, V. Tirronen, Recent advances in Differential Evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[100] N. Noman, H. Iba, Accelerating Differential Evolution using an adaptive local search, IEEE Trans. Evolut. Comput. 12 (1) (2008) 107–125.
[101] M. Olguin-Carbajal, E. Alba, J. Arellano-Verdejo, Micro differential evolution with local search for high dimensional problems, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC2013), 2013, pp 48–54.
[102] M.N. Omidvar, X.D. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evolut. Comput. 18 (3) (2014) 378–393.
[103] M.G.H. Omran, A.P. Engelbrecht, A. Salman, Bare bones Differential Evolution, Eur. J. Oper. Res. 196 (1) (2009) 128–139.
[104] Q.K. Pan, P.N. Suganthan, L. Wang, L. Gao, R. Mallipeddi, A Differential Evolution algorithm with self-adapting strategy and control parameters, Comput. Oper. Res. 38 (2011) 394–408.
[105] S.Y. Park, J.J. Lee, An efficient Differential Evolution using speeded-up k-nearest neighbor estimator, Soft Comput. 18 (1) (2014) 35–49.
[106] F. Peng, K. Tang, G.L. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Trans. Evolut. Comput. 14 (5) (2010) 782–800.
[107] F. Penunuri, C. Cab, O. Carvente, M.A. Zambrano-Arjona, J.A. Tapia, A Study of the classical Differential Evolution control parameters, Swarm Evolut. Comput. 26 (2016) 86–96.
[108] A.P. Piotrowski, J.J. Napiorkowski, Grouping Differential Evolution algorithm for multi-dimensional optimization problems, Control. Cybern. 39 (2) (2010) 527–550.
[109] A.P. Piotrowski, J.J. Napiorkowski, A. Kiczko, Differential Evolution algorithm with separated groups for multi-dimensional optimization problems, Eur. J. Oper. Res. 216 (2012) 33–46.
[110] A.P. Piotrowski, J.J. Napiorkowski, A. Kiczko, Corrigendum to: "Differential evolution algorithm with separated groups for multi-dimensional optimization problems" [Eur. J. Oper. Res. 216 (2012) 33–46], Eur. J. Oper. Res. 219 (2012) 488.
[111] A.P. Piotrowski, Adaptive Memetic Differential Evolution with global and local neighborhood-based mutation operators, Inf. Sci. 241 (2013) 164–194.
[112] A.P. Piotrowski, Differential Evolution algorithms applied to Neural Network training suffer from stagnation, Appl. Soft Comput. 21 (2014) 382–406.
[113] A.P. Piotrowski, Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions, Inf. Sci. 297 (2015) 191–201.
[114] I. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Inf. Sci. 297 (2015) 216–235.
[115] P. Posik, W. Huyer, L. Pal, A comparison of global search algorithms for continuous black box optimization, Evolut. Comput. 20 (4) (2012) 509–541.
[116] K.V. Price, An introduction to Differential Evolution, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 79–108.
[117] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution. A practical approach to global optimization, Springer-Verlag, Berlin-Heidelberg, 2005.
[118] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential Evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[119] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based Differential Evolution, IEEE Trans. Evolut. Comput. 12 (1) (2008) 64–79.
[120] S. Rahnamayan, H.R. Tizhoosh, Image thresholding using micro oppositionbased differential evolution (micro-ODE), in: Proceedings of IEEE World Congress on Computational Intelligence (WCCI-2008), Hong Kong, 2008, pp. 1409–1416.
[121] J. Rönkkönen, S. Kukkonen, K.V. Price, Real-parameter optimization with Differential Evolution. In: Proceedings of IEEE International Conference on Evolutionary Computation, vol. 1, 2005, pp. 506–513.
[122] M. Saraswat, K.V. Arya, H. Dharma, Leukocyte segmentation in tissue images using differential evolution algorithm, Swarm and Evolutionary Computation, , 11 (2013) 46–54.
[123] S. Sarkar, S. Das, S.S. Chaudhure, Hyper-spectral image segmentation using Rényi entropy based multi-level thresholding aided with Differential Evolution, Expert Syst. Appl. 50 (2016) 120–129.
[124] R.A. Sarker, S.M. Elsayed, T. Ray, Differential Evolution with dynamic parameters selection for optimization problems, IEEE Trans. Evolut. Comput. 18 (5) (2014) 689–707.
[125] S.C. Satapathy, A. Naik, Modified Teaching–Learning-based optimization algorithm for global numerical optimization—a comparative study, Swarm Evolut. Optim. 16 (2014) 28–37.
[126] C. Schumacher, M.D. Vose, L.D. Whitley, The no free lunch and problem description length, in: Proceedings of Genet. Evolut. Comput. Conf., 2001, pp. 565–570.
[127] K. Sörensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (2015) 3–18.
[128] R. Storn, K.V. Price, Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Tech. Report TR-95-012, International Computer Sciences Institute, Berkeley, California, USA, 1995.
[129] R. Storn, K.V. Price, Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[130] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, IIT Kanpur, India, 2005.
[131] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: 2014 Proceedings of IEEE Congress on Evolutionary Computation, 2014, pp. 1658–1665.
[132] K. Tang, F. Peng, G.L. Chen, X. Yao, Population-based algorithm portfolios with automated constituent algorithms selection, Inf. Sci. 279 (2014) 94–104.
[133] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, O. Buyukdagli, A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem, Comput. Oper. Res. 40 (2014) 1729–1743.
[134] J. Teo, Exploring dynamic self-adaptive populations in Differential Evolution, Soft Comput. 11 (8) (2006) 673–686.
[135] N.S. Teng, J. Teo, M.H.A. Hijazi, Self-adaptive population sizing for a tune-free Differential Evolution, Soft Comput. 13 (7) (2009) 709–724.
[136] V. Tirronen, F. Neri, Differential Evolution with fitness diversity self-adaptation, in: Raymond Chiong (Ed.), Nature-Inspired Algorithms for Optimisation, SCI 193, Springer-Verlag, Berlin Heidelberg, 2009, pp. 199–234.
[137] A. Trivedi, D. Srinivasan, S. Biswas, T. Reindl, Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem, Swarm and Evolutionary Computation, , 23 (2015) 50–64.
[138] J. Tvrdik, Adaptation in differential evolution: A numerical comparison, Appl. Soft Comput. 9 (3) (2009) 1149–1155.
[139] J. Tvrdık, R. Polakova, Competitive differential evolution applied to CEC 2013 problems, in: 2013 Proceedings of IEEE Congress on Evolutionary Computation, 2013, pp. 1651–1657.
[140] A. Ulas, O.T. Yildiz, E. Alpaydin, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recognit. 45 (2012) 1772–1781.
[141] O. Urfalioglu, O. Arikan, Self-adaptive randomized and rank-based Differential Evolution for multimodal problems, J. Glob. Optim. 51 (4) (2011) 607–640.
[142] C. Voglis, K.E. Parsopoulos, D.G. Papageorgiou, I.E. Lagaris, M.N. Vrahatis, MEMPSODE: a global optimization software based on hybridization of population-based algorithms and local searches, Comput. Phys. Commun. 183 (5) (2012) 1139–1154.
[143] C. Voglis, P.E. Hadjidoukas, K.E. Parsopoulos, D.G. Papageorgiou, I.E. Lagaris, M.N. Varhatis, p-MEMPSODE: Parallel and irregular memetic global optimization, Comput. Phys. Commun. 197 (2015) 190–211.
[144] J.A. Vrugt, B.A. Robinson, J.M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evolut. Comput. 13 (2) (2009) 243–259.
[145] H. Wang, S. Rahnamayan, Z.J. Wu, Adaptive Differential Evolution with variable population size for solving high-dimen-sional problems, in: Proceedings of IEEE Congress on Evolutionary Computation, 2011, pp 2626–2632.
[146] Y. Wang, Z.X. Cai, Q.F. Zhang, Differential Evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evolut. Comput. 15 (1) (2011) 55–66.
[147] C. Wang, J.H. Gao, High-dimensional waveform inversion with cooperative coevolutionary Differential Evolution algorithm, IEEE Geosci. Remote. Sens. Lett. 9 (2) (2012) 297–301.
[148] Y. Wang, Z.X. Cai, Q.F. Zhang, Enhancing the search ability of Differential Evolution through orthogonal crossover, Inf. Sci. 185 (1) (2012) 153–177.
[149] X. Wang, S.G. Zhao, Differential Evolution algorithm with self-adaptive population resizing mechanism, Math. Probl. Eng. (2013) 419372http://dx.doi.org/10.1155/2013/419372.
[150] Y. Wang, J. Huang, W.S. Dong, J.C. Yan, C.H. Tian, M. Li, W.T. Mo, Two-stage based ensemble optimization framework for large-scale global optimization, Eur. J. Oper. Res. 228 (2) (2013) 308–320.
[151] H. Wang, S. Rahnamayan, S. Hui, M.G.H. Omran, Gaussian bare bones Differential, Evol. IEEE Trans. Cybern. 43 (2) (2013) 634–647.
[152] Y. Wang, H.X. Li, T.W. Huang, L. Li, Differential Evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.

[153] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential Evolution enhanced with multi-objective sorting-based mutation operators, IEEE Trans. Cybern. 44 (12) (2014) 2792–2805.
[154] H. Wang, W.J. Wang, Z.H. Cui, H. Sun, S. Rahnamayan, Heterogeneous Differential Evolution for numerical optimization, Scientific World Journal (2014) 318063.
[155] M. Weber, F. Neri, V. Tirronen, Distributed Differential Evolution with explorative-exploitative population families, Genet. Program. Evol. Mach. 10 (4) (2009) 343–371.
[156] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed Differential Evolution, Soft Comput. 14 (2010) 1187–1207.
[157] M. Weber, F. Neri, V. Tirronen, Shuffle or update parallel Differential Evolution for large-scale optimization, Soft Comput. 15 (2011) 2089–2107.
[158] M. Weber, F. Neri, V. Tirronen, A study on scale factor in distributed Differential Evolution, Inf. Sci. 181 (12) (2011) 2488–2511.
[159] M. Weber, F. Neri, V. Tirronen, A study on scale factor/crossover interaction in distributed Differential Evolution, Artif. Intell. Rev. 39 (3) (2013) 195–224.
[160] D. Whitley, J. Rowe, Focused no free lunch theorems, in: Proceedings of Genet. Evol. Comput. Conf., 2008, pp. 811–818.
[161] D.H. Wolpert, W.G. Macready, No Free Lunch Theorems for Search, Tech. Rep. SFI-TR-05-010, Santa Fe, NM, USA, 1995.
[162] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evolut. Comput. 1 (1) (1997) 67–82.
[163] G.H. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H.G. Chen, Differential Evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[164] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing Differential Evolution and Particle Swarm Optimization to design powerful optimizers: a review and taxonomy, IEEE Trans. Syst. Mac Cybern. Part C – Appl. Rev. 42 (5) (2012) 744–767.
[165] M. Yang, C.H. Li, Z.H. Cai, J. Guan, Differential Evolution with auto-enhanced population diversity, IEEE Trans. Cybern. 45 (2) (2015) 302–315.
[166] W.J. Yu, M. Shen, W.N. Chen, Z.H. Zhan, Y.J. Gong, Y. Lin, O. Liu, J. Zhang, Differential Evolution with two-level parameter adaptation, IEEE Trans. Cybern. 44 (7) (2014) 1080–1099.
[167] D. Zaharie, Influence of crossover on the behavior of Differential Evolution algorithms, Appl. Soft Comput. 9 (3) (2009) 1126–1138.
[168] A. Zamuda, J. Brest, E. Menzura-Montes, Structured population size reduction Differential Evolution with multiple mutation strategies on CEC 2013 real parameter optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2013, pp. 1925–1931.
[169] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in Differential Evolution, Swarm Evolut. Comput. 25 (2015) 72–99.
[170] M. Zhang, W. Luo, X.F. Wang, Differential Evolution with dynamic stochastic selection for constrained optimization, Inf. Sci. 178 (15) (2008) 3043–3074.
[171] J. Zhang, Z.C. Sanderson, JADE: Adaptive Differential Evolution with optional external archive, IEEE Trans. Evolut. Comput. 13 (5) (2009) 945–958.
[172] S.Z. Zhao, P.N. Suganthan, Empirical investigations into the exponential crossover of Differential Evolutions, Swarm Evolut. Comput. 9 (2013) 27–36.
[173] S.G. Zhao, X. Wang, L. Chen, W. Zhu, A novel self-adaptive Differential Evolution algorithm with population size adjustment scheme, Arab. J. Sci. Eng. 39 (8) (2014) 6149–6174.
[174] Y.J. Zheng, H.V. Ling, H.H. Shi, H.S. Chen, S.Y. Chen, Emergency railway wagon scheduling by hybrid biogeography-based optimization, Comput. Oper. Res. 43 (2014) 1–8.
[175] W. Zhu, Y. Tang, J.A. Fang, W.B. Zhang, Adaptive population tuning scheme for Differential Evolution, Inf. Sci. 223 (2013) 164–191.