Swarm Intell (2016) 10:307–353
DOI 10.1007/s11721-016-0129-y

Searching for structural bias in particle swarm optimization and differential evolution algorithms

Adam P. Piotrowski[1] · **Jaroslaw J. Napiorkowski**[1]

Received: 17 November 2015 / Accepted: 1 November 2016 / Published online: 14 November 2016

Abstract During the last two decades, a large number of metaheuristics have been proposed, leading to various studies that call for a deeper insight into the behaviour, efficiency and effectiveness of such methods. Among numerous concerns that are briefly reviewed in this paper, the presence of a structural bias (i.e. the tendency, not justified by the fitness landscape, to visit some regions of the search space more frequently than other regions) has recently been detected in simple versions of the genetic algorithm and particle swarm optimization. As of today, it remains unclear how frequently such a behaviour occurs in population-based swarm intelligence and evolutionary computation methods, and to what extent structural bias affects their performance. The present study focuses on the search for structural bias in various variants of particle swarm optimization and differential evolution algorithms, as well as in the traditional direct search methods proposed by Nelder–Mead and Rosenbrock half a century ago. We found that these historical direct search methods are structurally unbiased. However, most tested new metaheuristics are structurally biased, and at least some presence of structural bias can be observed in almost all their variants. The presence of structural bias seems to be stronger in particle swarm optimization algorithms than in differential evolution algorithms. The relationships between the strength of the structural bias and the dimensionality of the search space, the number of allowed function calls and the population size are complex and hard to generalize. For 14 algorithms tested on the CEC2011 real-world problems and the CEC2014 artificial benchmarks, no clear relationship between the strength of the structural bias and the performance of the algorithm was found. However, at least for artificial benchmarks, such old and structurally unbiased methods like Nelder–Mead

Electronic supplementary material The online version of this article (doi:10.1007/s11721-016-0129-y) contains supplementary material, which is available to authorized users.

✉ Adam P. Piotrowski
adampp@igf.edu.pl

Jaroslaw J. Napiorkowski
jnn@igf.edu.pl

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

algorithm performed relatively well. This is a warning that the presence of structural bias in novel metaheuristics may hamper their search abilities.

Keywords Metaheuristic · Differential evolution · Particle swarm optimization · No-free-lunch theorem · Structural bias · Search space

1 Introduction

Although a number of direct search methods (Kolda et al. 2003) to solve continuous optimization problems were already proposed over half a of century ago, the last two decades have witnessed a very rapid influx of papers introducing novel metaheuristics (e.g. Boussad et al. 2013; Xing and Gao 2014; Salcedo-Sanz 2016). A vast majority of papers propose yet another optimization algorithm. A much smaller fraction of papers aims at understanding the behaviour of these algorithms. However, in recent years important questions regarding the behaviour and usefulness of the metaheuristics as well as the quality of the respective research (both in general terms and with regard to specific methods) have been raised.

The potential deficiency of metaheuristics, referred to as *structural bias*, that is discussed in this study has been addressed by Kononova et al. (2015). According to Kononova et al. (2015), a heuristic algorithm is structurally biased when it is more prone to visit some parts of the search space than the other parts and this behaviour is not justified by the fitness landscape. They found such a tendency in a simple variant of the genetic algorithm (GA) (Holland 1975) and a basic particle swarm optimization (PSO) algorithm (Eberhart and Kennedy 1995). As metaheuristics are designed to sample the search space based only on the fitness function's values observed in already visited points, such behaviour may suggest that the algorithm itself is somehow biased. As a very simple example, imagine a heuristic that moves a population of individuals in the search space according to the $x_{ij}^{t+1} = x_{ij}^{t}/rand(1, 2)$ rule, where x_{ij}^{t} is the position of the ith individual in the jth dimension at the tth iteration, and $rand(1, 2)$ is a random number generated from the uniform distribution within the interval [1, 2]. Such an algorithm, composed of individuals that move independently, is structurally biased and will converge to $\mathbf{x} = 0$. The closer to $\mathbf{x} = 0$ the optimum of a particular problem is, the higher the chance that such an algorithm finds it. Because in many artificial benchmark functions used to test optimizers the optimum is located in, or close to $\mathbf{x} = 0$ [as for example in 40% of problems defined in the Yao et al. (1999) paper], such a naive procedure defined above could seemingly be very successful. This is of course a trivial example, but if an algorithm is more complicated, the presence of some structural bias may be easily overlooked. This will lead to doubtful conclusions of good performance, especially if an approach is tested on a small number of simple problems.

The presence of structural bias may be determined by testing algorithms on a fully random function $f(\boldsymbol{x})$ that maps any location $\boldsymbol{x}$ to uniformly random values from a uniform distribution within [0, 1]. Values of $f(\boldsymbol{x})$ depend neither on the values of f in the neighbourhood of $\boldsymbol{x}$ nor on the values of f sampled in the past. Without loss of generality, minimization of f is considered in this study. Examples of such functions have been given in Cleghorn and Engelbrecht (2014, 2015) and Kononova et al. (2015). In this study, we use the variant of the function proposed in Kononova et al. (2015):

$$f : S \subset R^D \rightarrow [0, 1], \qquad f(\boldsymbol{x}) \in uniform[0, 1], \qquad S = [0, 1]^D \tag{1}$$

The function defined in Eq. (1) does not have any stable fitness landscape, as each time a particular location $\boldsymbol{x}$ is visited, the value of $f(\boldsymbol{x})$ is generated anew from the uniform distribution.

This is a slightly different approach from that proposed in Cleghorn and Engelbrecht (2014, 2015), where f is static during a particular run. Hence, when searching for the minimum of f, if the algorithm does not sample some parts of the search space more often than the others (due to algorithms' own structural bias), the distribution of the positions of the best solutions found during multiple runs should also be uniform in the search space. This can be confirmed by a visual inspection of the respective plots or, more objectively, by using some statistical tests. If this distribution cannot be considered uniform, there must be some structural bias of the algorithm itself.

In Kononova et al. (2015), it is suggested that the structural bias is "stronger" when the population size increases and that such bias has higher impact on the search efficiency when the problem to be solved is "difficult". However, Kononova et al. (2015) examined only two very simple metaheuristics, basic variants of GA and PSO, and just on 30-dimensional variants of f defined in Eq. (1), with the number of function calls fixed to 300,000. Each experiment was repeated 50 times. Although this value is frequently used for statistical tests, to convincingly detect structural bias in specific algorithms a much larger number of repetitions may be needed. In addition, in Kononova et al. (2015) more attention was paid to visual observations than statistical tests, even though such observations are inevitably subjective.

The present study expands the research described in Kononova et al. (2015) and goes a step further. The main goal is to search for the presence of structural bias in various variants of two popular families of metaheuristics, namely PSO and differential evolution (DE) (Storn and Price 1997), and in two widely used historical direct search methods (Nelder and Mead 1965; Rosenbrock 1960). In addition, we verify how quickly the effect of the structural bias can be observed and how strong are the impacts of the problem's dimensionality and algorithm's population size on the occurrence of structural bias. We also find the source of structural bias in one selected variant of PSO and show that it may be removed. Finally, all 14 algorithms that are tested for the presence of structural bias are also used to solve 22 real-world problems from CEC2011 (Das and Suganthan 2010) and 50-dimensional versions of 30 artificial benchmarks from CEC2014 (Liang et al. 2013). The objective is to check whether the presence of structural bias can indeed affect the practical application of metaheuristics (see discussion in Michalewicz 2012).

2 Background: recent criticisms of optimization metaheuristics

Notwithstanding the wide popularity of metaheuristics and their overall acclamation shared by the majority of papers devoted to these methods, some doubts have been raised in recent literature. Although in this study we pay attention mostly to the problem of structural bias in the algorithms (Kononova et al. 2015), it is useful to place our discussion in a broader context. First of all, the abundance of metaheuristics that are frequently developed without any theoretical justification leads to at least three undesired effects: (i) it is hard to properly choose the right method for a particular task (Muñoz et al. 2015; Yuen and Zhang 2015); (ii) although performance of some widely used metaheuristics is promising, many others show little (if any) novelty and efficiency—instead they rather compete for popularity by using appealing nomenclature (Sörensen 2015); (iii) even though studies of the behaviour of metaheuristics do appear (Clerc and Kennedy 2002; Van den Bergh and Engelbrecht 2004; Auger and Doerr 2011; Liao et al. 2013; Bonyadi and Michalewicz 2014; Cleghorn and Engelbrecht 2014; Hu et al. 2014; Rada-Vilela et al. 2014; Leonard et al. 2015; Hu et al. 2016), in the majority of papers a desire to propose yet another novel optimizer dominates

over the willingness to gain deeper insight into the already available methods. This fact, combined with a lack of commonly accepted procedures to properly develop, study and compare metaheuristics [see for example the discussion in Michalewicz 2012 and Sörensen et al. 2015], triggered a number of critical research outputs that identified many important issues.

From the no-free-lunch (NFL) theorems for optimization, that have already been widely discussed from various points of view (Wolpert and Macready 1997; Köppen et al. 2001; Schumacher et al. 2001; Droste et al. 2002; Whitley and Rowe 2008; Rowe et al. 2009; Duéñez-Guzmán and Vose 2013), it is well known that the performance of all heuristic optimizers averaged over all problems is equal. Although it is generally understood that only a very small fraction of "all problems" [for a detailed discussion of what "all problems" means, see Wolpert and Macready (1997) and Schumacher et al. (2001)] is of interest to anyone (Igel and Toussaint 2003, 2004), it is very unfortunate that the implications of the NFL are rarely considered and discussed when novel metaheuristics are proposed. Although any applicable set of problems would be just an extremely small fraction of "all problems", this is not an excuse for the fact that the new methods are frequently tested on very few artificial benchmark problems. Moreover, such benchmark problems are often the ones for which the performance of the promoted algorithms is good enough to justify their being proposed. Although relationships between the types of problems, or their specific features, and the performance of algorithms may be identified (Malan and Engelbrecht 2013, 2014; Muñoz and Smith-Miles 2016), such relations are very rarely studied in practice. As a result, when metaheuristics are applied to solve practical tasks, despite their popularity and wide acclamation (Fogel 2000; Eiben and Smith 2015; Zelinka 2015), they are often outperformed by problem-specific algorithms (Droste et al. 2006), or, when computational budget is limited, mathematical programming approaches (Pošik et al. 2012). Moreover, although the convergence of most metaheuristics cannot be proven, one may for some of them prove that on specific types of problems the convergence of such methods cannot be guaranteed (Hu et al. 2016).

Numerous studies empirically show how disputable the empirical comparison of metaheuristics may be. For example, Derrac et al. (2014) noted in the research based on the CEC2011 real-world problems (Das and Suganthan 2010) that very different rankings of metaheuristics may be obtained depending on whether statistical tests are used to verify the statistical significance of differences between the final results achieved by the different metaheuristics versus when the tests are used to verify the differences in the convergence properties of such metaheuristics. Also Veček et al. (2014) emphasize the extent to which the final opinion on the algorithms' performance may depend on the choice of the statistical tests used. In Dymond et al. (2015), it was pointed out that the choice of proper control parameter values for various metaheuristics largely depends on the number of function calls used. Liao et al. (2013) showed that the claimed superiority of the newer variants of artificial bee colony (ABC) algorithms over the older methods may be confirmed when tests are performed on simple low-dimensional problems, but not when more difficult high-dimensional problems are considered. Draa (2015) showed, based on a wide selection of problems, that the performance of the flower pollination algorithm, one of the recently introduced approaches, turns out to be relatively poor when compared with classical algorithms, in contradiction to what is claimed in the literature. Similarly, Fong et al. (2016) found the lack of superiority of a novel bat-inspired algorithm over the classical PSO. Finally, Piotrowski (2015) and Piotrowski and Napiorkowski (2016) showed that very different algorithms rankings may be derived when one performs tests either by searching for the minimum versus the maximum of exactly the same functions.

The above discussion shows that the way in which comparative analysis is conducted may affect the opinion on the performance of competing metaheuristics—the results may even be skewed to draw the intended conclusions. No wonder then that the very quality of research reported in the papers analysing metaheuristics has been questioned a number of times. Mernik et al. (2015) studied all papers on artificial bee colony (ABC) that were available up to August 2013 and referred to the first ABC algorithm published in the journal paper (Karaboga and Basturk 2008). They found that, depending on the evaluation criteria, between 40 and 67% of such studies led to results that are not trustworthy or even simply wrong, due to improperly made comparisons among different methods. Some papers (Ampellio and Vassio 2016) introduce both the novel algorithm and the novel method of comparison between algorithms, which may lead to some doubts regarding the evaluation of the achieved results. Črepinšek et al. (2016) discussed in depth how, by manipulating the replications of experiments, the comparison of metaheuristics may become unfair. This problem of a frequent lack of scrutiny, that may lead to erroneous results when comparing various metaheuristics, has also been addressed in numerous other studies (Eiben and Jelasity 2002; Weise et al. 2012; Črepinšek et al. 2012, 2014; Chinta et al. 2016). The fact that similar problems are also observed in other fields of science (Yancey 1990; Hall et al. 1996; Kitchenham et al. 2002) should not be considered as an excuse.

Apart from the difficulty of setting fair comparison rules, and occasional errors and mistakes, the simple lack of originality in some quasi-novel metaheuristics has been discussed in various recent papers. It seems that in many recently proposed methods the novelty is largely… the name. For example, Weyland (2010, 2015) showed that the harmony search algorithm is just a special case of one of the well-known evolutionary strategies (Rechenberg 1973). Piotrowski et al. (2014) pointed out that the black-hole-based heuristic algorithm is just a much simplified variant of PSO with inertia weight (Shi and Eberhart 1998). Simon et al. (2011) discussed the large similarities between biogeography-based optimization and many older metaheuristics. A number of recently introduced physical processes-based approaches are also considered to include mainly replications of the former methods (Salcedo-Sanz 2016), and a few core components that are widely used in various metaheuristics are identified in Fong et al. (2016) to expose the lack of novelty in some of the most recent methods. According to Sörensen (2015), introducing such redundant approaches can be partly explained by the common practice of hiding the lack of methodological novelty behind the new nomenclature that comes from the "inspiration"—resulting in algorithms based on "quantum frogs", "competitive imperialists" or "intelligent water drops". A kind of recipe of how to prepare such algorithms is outlined by Fister et al. (2016) to draw more attention to the problem. Luckily, authors of some papers (Sörensen et al. 2015; Fister et al. 2016) argue that there are signs that such a tendency would soon be stopped. As one may conclude from this brief review, the question regarding the presence of structural bias in such popular families of optimization algorithms as particle swarm optimization and differential evolution just touches the tip of the iceberg that, if warmed up, may flood the metaheuristics with at least partly-deserved scepticism.

3 Methods for structural bias detection

In this paper, the 14 metaheuristics and direct search methods listed in Table 1 are tested for the presence of structural bias. All abbreviations of the specific algorithms are defined in Table 1. Among these 14 methods, there are five PSO (PSO-basic—the first version proposed by Eberhart and Kennedy (1995), ALC-PSO, CLPSO, DNS-PSO and PSO-init-

Table 1 Algorithms tested for the presence of structural bias

Abbreviation	Full name	References	Comments
PSO-basic	Particle swarm optimization (first version)	Eberhart and Kennedy (1995)	The initial particle velocities are randomly generated within 20% of the bounds span. During run velocities are also restricted to be not higher than 20% of the bounds span. The global version of the model is used, as it was proposed first in Eberhart and Kennedy (1995). $c1 = c2 = 2$, as suggested in Kennedy and Eberhart (1995), even though this choice was later dismissed by Clerc and Kennedy (2002)
ALC-PSO	PSO with an ageing leader and challengers	Chen et al. (2013)	The initial particle velocities are set to 0. Velocities are restricted to be not larger than 50% of the bounds span. Control parameters are set according to Chen et al. (2013) as follows: $c1 = c2 = 2$, lifespan $\Theta_0 = 60$, static inertia weight $\omega = 0.4$, $T = 2$, $pro = 1/D$ (D is the problem dimensionality)
CLPSO	Comprehensive learning PSO	Liang et al. (2006)	The source code has been obtained from co-author of Liang et al. (2006). Velocities are restricted to be not larger than 20% of the bounds span. The initial PSO velocities are generated randomly from uniform distribution within 20% of the bounds span. Control parameters are as follows: $c = 1.49445$, inertia weight decreases linearly during run from 0.9 to 0.2
DNS-PSO	Diversity-enhanced PSO with neighbourhood search	Wang et al. (2013)	Velocities are restricted to be not larger than 50% of the bounds span. The initial velocities are generated randomly from uniform distribution within 50% of the bounds span. Control parameters are as follows (Wang et al. (2013)): $c1 = c2 = 1.49618$, $w = 0.7298$, $pr = 0.9$, $pns = 0.6$, $k = 2$
PSO-init-weight	PSO with inertia weight	Shi and Eberhart (1998)	The initial particle velocities are randomly generated within 20% of the bounds span. During run velocities are also restricted to be not higher than 20% of the bounds span. $c1 = c2 = 1.49$, w decreases linearly from 0.9 to 0.4 during run
DE-basic	Differential evolution (first version)	Storn and Price (1995)	The simplest version from the first report is used here (with scheme called there DE1 that uses the mutation strategy known in later papers by DE/rand/1), with $F = 0.8$, $Cr = 0.5$ (also considered in Storn and Price 1995)
CDE	Clustering DE	Cai et al. (2011)	The first DE algorithm based on clustering that uses DE/rand/1/exp variant of mutation and crossover. Control parameters are set as follows (Cai et al. 2011): $F = 0.5$, $Cr = 0.9$, clustering period $m = 10$
DEGL	DE with global and local neighbourhood mutation operators	Das et al. (2009)	DEGL variant with self-adaptive weight factor is used (DEGL/SAW), as suggested in Das et al. (2009). Control parameters are as follows (Das et al. 2009): $F = 0.8$, $Cr = 0.9$, neighbourhood size $k = 5\%$ (such that total number of individuals in the neighbourhood defined on the ring topology equals $2k = 10\%$ of *PopSize*; note, however, that in our application k cannot be lower than 2)

Table 1 continued

Abbreviation	Full name	References	Comments
MDE-pBX	Modified DE with p-best crossover	Islam et al. (2012)	The source code has been obtained from co-author of Islam et al. (2012). Control parameters are set as follows (Islam et al. 2012): $q\% = 0.15 \cdot PopSize$, n (in power mean calculation) $= 1.5$; Fm is initialized to 0.5; Crm is initialized to 0.6
SADE	Self-adaptive DE	Qin et al. (2009)	One of the most popular self-adaptive evolutionary algorithm. Variant with four strategies proposed in Qin et al. (2009) is used. Control parameters are set as follows: learning period $LP = 40$, F is generated from normal distribution $N(0.5, 0.3)$, Cr is generated from normal distribution $N(Crm, 0.1)$, when Crm is initialized to 0.5 and varies during search (see Qin et al. 2009), $\varepsilon = 0.01$
CLPSO-DEGL	CLPSO-DEGL hybrid algorithm	Epitropakis et al. (2012)	The source code of CLPSO part has been obtained from co-author of Liang et al. (2006). DEGL variant with self-adaptive weight values is used, as suggested in Das et al. (2009). The same control parameters as in CLPSO and DEGL are used
GA-MPC	Genetic algorithm with multi-parent crossover	Elsayed et al. (2011, 2014)	The source code has been obtained from CEC2011 web page, as GA-MPC was the winner of the competition. According to Elsayed et al. (2014), the 2014 version differs just in constraints handling. GA-MPC has large number of control parameters for which we refer the reader to Elsayed et al. (2011)
NMA	Nelder–Mead simplex	Nelder and Mead (1965)	The variant based on Lagarias et al. (1998) is used. When solving benchmark problems, a re-initialization approach defined as NMA1 in Piotrowski (2015) has been used. All solutions are re-initialized randomly (without keeping the best one) when two criteria are met: 1. the maximum difference between coordinates of the best and the worst point is lower than 10^{-4}; 2. difference in fitness between the best and the worst point is lower than 10^{-4}. Following control parameters are used: $\alpha = 1, \gamma = 2, \beta = \sigma = 0.5$
RA	Rosenbrock's algorithm	Rosenbrock (1960)	The following parameter values are used: $\alpha = 3$, $\beta = -0.5$, initial step $= 0.1$ (variable during search). When solving benchmark problems, a re-initialization approach defined as RA2 in Piotrowski (2015) has been used. It works as follows: after every $100D$ function calls, it is verified whether during last $100D$ function calls the solution was improved by more than 10^{-4}. If it does not happen, the location of RA point is randomly re-initialized, the value of step is reset to 0.1 and the coordinates are reset to initial system

weight) and five DE variants (DE-basic—the first version proposed by Storn and Price in 1995, CDE, DEGL, MDE_pBX and SADE), one hybrid PSO-DE approach (CLPSO-DEGL), two historical direct search methods (NMA and RA) and, for comparison, the winner of the CEC2011 competition (GA-MPC). Control parameters of algorithms are mainly the same as in the original papers, with the exception of the population sizes and of those parameters discussed in the comments column of Table 1. Using standard values of control parameters is important, as different control parameter values may lead to exaggeration or mitigation of the presence of structural bias. We explicitly verify the impact of the population size settings on the structural bias in various metaheuristics. Apart from CLPSO and GA-MPC, which implement their own approaches to handling boundary constraints (Liang et al. 2006; Elsayed et al. 2011), the reflection method (Helwig et al. 2013) is used to prevent algorithms from exceeding bounds.

The PSO and DE algorithms are chosen in this study due to their large popularity and versatile applications in various fields of science. The interested reader may find details on both families of methods in numerous reviews or books (Price et al. 2005; Clerc 2006; Poli et al. 2007; Banks et al. 2008; Das et al. 2008, 2016; Neri and Tirronen 2010; Das and Suganthan 2011; Xin et al. 2012; Rada-Vilela et al. 2014; Zhang et al. 2015; Piotrowski 2016; Bonyadi and Michalewicz 2016).

To find the presence of structural bias, each algorithm is tested in numerous experiments on the fully random function defined in Eq. (1) with the following conditions:

- the dimension (D) of the domain of function f (Eq. (1)) is set to 2, 10, 30, 100 and 1000;
- the maximum number of allowed function calls (MNFC) is set to 10,000 and 300,000;
- the population size (*PopSize*) of each algorithm, except for RA and NMA, is set to 20, 50, 100 and 500 (by definition, RA, which is the only non-population-based heuristic chosen to be used in this study, may only be tested with $PopSize = 1$, and for NMA the *PopSize* is always related to D by $PopSize = D + 1$).

Hence, for each algorithm $5 \cdot 2 \cdot 4 = 40$ experiments are performed (with the exception of RA and NMA, for which we have $5 \cdot 2 \cdot 1 = 10$ experiments). Each experiment is repeated 1000 times to achieve a statistically meaningful sample. The locations of the best solution found after random initialization of the population (i.e. after *PopSize* function calls) and the best solution found after MNFC are stored. Samples that contain 1000 elements are considered large enough to detect the presence of structural bias and to relate its strength to the problem dimensionality, the number of allowed function calls or the population size of the algorithm.

To check for the presence and the strength of structural bias, the one-sample and the two-sample Kolmogorov–Smirnov (KS) statistical tests (Stephens 1970; Press et al. 2006), of which only the first has been used in the previous study by Kononova et al. (2015), are applied to each D-dimensional sample of 1000 best locations found in each experiment. As in Kononova et al. (2015), each test is performed independently for each dimension, giving D results (p values) for every sample.

The one-sample KS test verifies whether the empirical distribution function (edf) of the sample of locations of the best solutions found within the MNFC for a particular dimension is different from the expected cumulative distribution function (cdf)—the uniform distribution in this case. Hence, the results depend on the coordinate system [i.e. not rotation invariant; see Salomon (1996)]. Hypothesis H0 is that the positions of the best individuals in a particular dimension found by the algorithm follow the uniform distribution. The H0 hypothesis is rejected at the $\alpha = 0.05$ significance level if the p value obtained in the test for a particular dimension is below 0.05. For every algorithm (and case considered), the number of such tests is equal to the problem dimensionality. The discussion is based on percentages of

rejected H0 hypotheses, the lowest (among D) obtained p values and—in the case of selected algorithms—graphical illustrations. It is assumed that the more frequently H0 hypotheses are rejected, the higher the structural bias of the particular algorithm is. Note that in Kononova et al. (2015), only 30-dimensional variants of f were tested, limiting the conclusions drawn there. For 1000-dimensional variants of f, it is expected that 50 out of 1000 tests (i.e. 5%) reject the true H0, but for the 30-dimensional case only 1.5 out of 30 tests performed are expected to reject the true H0. This is an important difference when higher rejection rates are considered as an effect of the presence of structural bias.

Some readers may suspect that the structural bias could be caused not by the algorithm itself, but by the pseudo-random number generator, though Kononova et al. (2015) claimed that this is not the case. All algorithms tested in this study were implemented in MATLAB (or obtained from their inventors; see notes in Table 1), using the Mersenne Twister (Matsumoto and Nishimura 1998) as the random number generator. To verify whether the bias is not present in the randomly initialized solutions (in such a case it would be hard to blame the algorithm), a one-sample KS test is used twice for each experiment, the second time to test the H0 that the edf of best solutions obtained just after initialization of the algorithm is from a uniform distribution. Then, the results of the latter are compared with the results of a one-sample KS test performed on the sets of the best solutions after the algorithm completed the MNFC.

In addition, in this study the two-sample KS test is also performed (again, separately for each dimension at the $\alpha = 0.05$ significance level). The two-sample KS test compares two empirical distribution functions—the edf of the sample of locations of the best solutions found within the MNFC and the edf of the sample of best solutions obtained right after initialization of the algorithm. The hypothesis H0 is that both samples come from the same distribution. The results of both one-sample and two-sample tests, performed for all algorithms, population sizes and problem dimensionalities considered, are discussed in Sect. 4.1.

The number of allowed function calls may affect the results obtained. In practical applications, MNFC is sometimes related linearly to the problem dimensionality [for example to $10{,}000D$, where D is the problem dimensionality, as in the CEC2005 problems introduced in Suganthan et al. (2005), or the CEC2014 benchmarks proposed by Liang et al. (2013)] or is fixed, being independent of the problem dimensionality [as in the collection of CEC2011 real-world problems introduced in Das and Suganthan (2010)]. Also, in various practical applications the MNFCs noticeably differ. It has already been shown by Pošik et al. (2012) that metaheuristics are mostly useful when the MNFC is relatively large; when it is small, mathematical programming methods are suggested. This is why we conducted our main experiments with a MNFC set to 10,000 and 300,000. However, as the present paper aims to also verify how quickly the presence of structural bias may be detected, apart from the experimental settings discussed above, additional tests are performed by running all algorithms 1000 times with *PopSize* set to 100 (apart from NMA and RA, which require specific values of *PopSize*) on the 1000-dimensional fully random function (Eq. (1)) for very low numbers of function calls, namely 200, 300, 400, 500, 1000, 2000 and 5000. Runs for each number of function calls are performed independently. The H0 hypothesis that the obtained empirical distributions of best solutions found within so few function calls follow the uniform distribution is then tested by the one-sample KS test. This allows us to show how quickly the presence of structural bias may be observed for different DE and PSO variants. Note that these additional experiments are limited to specific values of population size and problem dimensionality. The results are discussed in Sect. 4.2.

Finally, to check whether there is any relationship between the possible presence (and strength) of structural bias and the performance of the algorithm on various kinds of problems,

each tested heuristic is applied to solve 22 real-world problems from the CEC2011 set (Das and Suganthan 2010) and 50-dimensional versions of 30 artificial benchmarks from the CEC2014 set (Liang et al. 2013). Due to much longer computational time, for the CEC2011 and CEC2014 problems each algorithm is tested only with one selected population size that seems suitable for its practical application. The number of allowed function calls is set to 150000 for the CEC2011 problems (as suggested in Das and Suganthan 2010) and to 10,000D for the CEC2014 benchmark problems (as suggested by Liang et al. 2013). The number of runs is reduced to 51 (as suggested in Liang et al. 2013). The statistical significance of pairwise comparisons among all 14 algorithms is tested by means of the Friedman's test with post hoc Shaffer's static procedure (Shaffer 1986), as suggested in Garcia and Herrera (2008) and Ulas et al. (2012). The MATLAB implementation of Shaffer's procedure, developed by Ulas et al. (2012), and available at www.cmpe.boun.edu.tr/~ulas/m2test/details.php, is applied in this study.

4 Experiments

Discussion of the results is divided into four parts. Section 4.1 discusses the wide-scale search for the presence of structural bias in heuristic algorithms. Section 4.2 shows how quickly the effects of the structural bias may be observed for various algorithms in a case study experiment. Section 4.3 aims at finding the reason for the structural bias in a specific algorithm. Finally, in Sect. 4.4 the results achieved by all considered algorithms on the CEC2011 and 50-dimensional versions of the CEC2014 benchmark problems are discussed in order to determine the relationship between the strength of the structural bias and the algorithm's practical performance.

4.1 The presence of structural bias

The percentage of dimensions for which the presence of structural bias was detected by the one-sample KS test at $\alpha = 0.05$ (i.e. the H0 hypothesis was rejected) is given in Table 2. This number is shown for every combination of algorithm (columns), problem dimensionality, D (row), and population size, *PopSize* (row). The upper part of Table 2 displays the results obtained when MNFC was set to 10,000, whereas the lower part displays the results when MNFC was set to 300,000. Table 3 contains the respective percentages of rejections of H0 hypotheses for the best solutions found right after random initialization of the population, i.e. before the proper algorithm started to work. Table 4 presents the lowest p values obtained by the KS test among all D dimensions for each considered experimental setting with MNFC $=$ 10,000 (upper part) and MNFC $=$ 300,000 (lower part). Table 5 shows the lowest p values of KS tests performed for solutions obtained just after initialization, before the algorithms started. These experiments are carried out to find out whether some bias in locations of the best solutions is observed:

- immediately after initialization (the presence of such bias cannot be attributed to the algorithm);
- after the algorithm was run for a relatively small number of function calls set to 10,000;
- after the algorithm was run for a number of function calls set to a more frequently used value of 300,000;
- or such bias is not observed at all.

Table 2 Percentages of H0 hypotheses rejected by Kolmogorov–Smirnov test at $\alpha = 0.05$ significance level. The H0 hypothesis is that the positions of best individuals are from uniform distribution in the final generation, after 10,000 (upper part of the table) or 300,000 (lower part of the table) function calls. H^0 is tested for each among D dimensions separately. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
10,000 function calls															
2	20	100.0	100.0	100.0	100.0	100.0	0.0	0.0	50.0	50.0	0.0	100.0	100.0	0.0	100.0
2	50	100.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0	50.0	0.0	100.0	100.0		
2	100	100.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0	50.0	0.0	100.0	100.0		
2	500	100.0	50.0	100.0	100.0	100.0	0.0	0.0	0.0	100.0	0.0	100.0	100.0		
10	20	70.0	60.0	90.0	100.0	100.0	20.0	10.0	80.0	90.0	40.0	100.0	100.0	10.0	30.0
10	50	80.0	20.0	60.0	100.0	100.0	0.0	0.0	30.0	50.0	40.0	100.0	80.0		
10	100	90.0	40.0	50.0	100.0	100.0	0.0	0.0	80.0	80.0	20.0	90.0	90.0		
10	500	100.0	80.0	70.0	100.0	100.0	0.0	10.0	30.0	90.0	20.0	80.0	60.0		
30	20	53.3	13.3	30.0	100.0	100.0	10.0	10.0	46.7	66.7	73.3	100.0	100.0	0.0	20.0
30	50	96.7	40.0	33.3	100.0	100.0	6.7	3.3	46.7	40.0	53.3	100.0	86.7		
30	100	100.0	80.0	16.7	100.0	100.0	3.3	3.3	50.0	70.0	43.3	96.7	83.3		
30	500	100.0	70.0	10.0	100.0	100.0	0.0	6.7	46.7	93.3	36.7	93.3	76.7		
100	20	77.0	20.0	10.0	100.0	100.0	9.0	13.0	42.0	81.0	56.0	100.0	98.0	4.0	6.0
100	50	95.0	30.0	21.0	100.0	100.0	5.0	8.0	55.0	57.0	47.0	98.0	95.0		
100	100	96.0	72.0	17.0	100.0	100.0	9.0	4.0	41.0	56.0	39.0	97.0	83.0		
100	500	100.0	57.0	9.0	100.0	100.0	5.0	7.0	47.0	86.0	24.0	85.0	82.0		
1000	20	72.0	19.1	9.5	100.0	100.0	5.1	11.6	44.8	77.7	56.5	97.0	98.9	4.9	4.8
1000	50	94.9	32.7	8.8	100.0	100.0	5.2	7.2	56.2	56.2	52.0	98.3	91.5		
1000	100	99.2	64.3	9.3	100.0	100.0	4.6	4.5	44.8	54.4	40.1	93.6	80.4		
1000	500	99.8	64.4	8.7	100.0	100.0	4.7	5.0	43.9	86.1	28.9	84.4	69.3		

Table 2 continued

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
300,000 function calls															
2	20	50.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0	50.0	50.0	100.0	100.0	0.0	100.0
2	50	50.0	100.0	100.0	100.0	100.0	0.0	50.0	0.0	0.0	50.0	100.0	100.0		
2	100	0.0	100.0	100.0	100.0	100.0	0.0	0.0	50.0	0.0	50.0	100.0	50.0		
2	500	50.0	50.0	100.0	100.0	100.0	0.0	0.0	50.0	0.0	50.0	100.0	100.0		
10	20	20.0	100.0	100.0	100.0	100.0	0.0	0.0	30.0	30.0	50.0	100.0	100.0	10.0	30.0
10	50	50.0	100.0	100.0	100.0	100.0	0.0	10.0	40.0	0.0	70.0	100.0	90.0		
10	100	80.0	100.0	100.0	100.0	100.0	10.0	0.0	10.0	30.0	20.0	100.0	80.0		
10	500	100.0	30.0	100.0	100.0	100.0	0.0	10.0	10.0	30.0	30.0	100.0	80.0		
30	20	33.3	60.0	60.0	100.0	100.0	6.7	20.0	36.7	26.7	73.3	100.0	100.0	6.7	13.3
30	50	56.7	80.0	60.0	100.0	100.0	6.7	6.7	16.7	20.0	73.3	100.0	93.3		
30	100	46.7	56.7	43.3	100.0	100.0	6.7	3.3	16.7	13.3	43.3	96.7	76.7		
30	500	83.3	46.7	26.7	100.0	100.0	6.7	10.0	20.0	20.0	60.0	96.7	83.3		
100	20	23.0	52.0	19.0	100.0	100.0	5.0	29.0	14.0	28.0	61.0	99.0	100.0	6.0	12.0
100	50	53.0	70.0	21.0	100.0	100.0	3.0	9.0	23.0	23.0	66.0	99.0	95.0		
100	100	63.0	66.0	25.0	100.0	100.0	6.0	6.0	13.0	16.0	57.0	92.0	93.0		
100	500	91.0	27.0	7.0	100.0	100.0	3.0	6.0	18.0	25.0	55.0	90.0	84.0		
1000	20	27.1	44.4	9.0	100.0	100.0	5.4	29.7	11.7	33.8	75.1	95.4	99.6	5.0	4.9
1000	50	54.3	59.8	9.3	100.0	100.0	5.3	13.6	25.1	24.0	63.1	96.1	92.7		
1000	100	66.9	47.4	8.3	100.0	100.0	5.4	12.0	14.6	19.5	62.7	89.3	86.9		
1000	500	91.6	34.7	9.3	100.0	100.0	5.2	4.6	24.2	18.8	58.5	88.6	82.9		

Table 3 Percentages of H0 hypotheses rejected by Kolmogorov–Smirnov test at $\alpha = 0.05$ significance level. The H0 hypothesis is that the positions of best individuals are from uniform distribution after initialization. As initialization for tests with 10,000 and 300,000 function calls is performed independently, results for both cases are presented separately. H0 is tested for each among D dimensions separately. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE- pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Initialization for tests with 10,000 function calls															
2	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0
2	50	0.0	0.0	100.0	0.0	50.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0		
2	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		
2	500	0.0	0.0	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0		
10	20	0.0	0.0	0.0	10.0	10.0	20.0	0.0	10.0	0.0	0.0	10.0	10.0	0.0	0.0
10	50	0.0	0.0	0.0	0.0	0.0	20.0	0.0	10.0	10.0	10.0	0.0	10.0		
10	100	10.0	0.0	10.0	20.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0		
10	500	0.0	0.0	40.0	0.0	0.0	30.0	0.0	0.0	0.0	0.0	30.0	30.0		
30	20	3.3	3.3	6.7	6.7	0.0	0.0	10.0	3.3	6.7	0.0	0.0	16.7	3.3	13.3
30	50	0.0	3.3	0.0	0.0	3.3	3.3	0.0	3.3	0.0	0.0	6.7	10.0		
30	100	6.7	0.0	6.7	6.7	6.7	3.3	0.0	13.3	0.0	3.3	3.3	20.0		
30	500	6.7	3.3	10.0	0.0	6.7	6.7	6.7	0.0	10.0	0.0	10.0	16.7		
100	20	3.0	7.0	7.0	6.0	9.0	4.0	4.0	7.0	5.0	2.0	10.0	9.0	6.0	5.0
100	50	6.0	7.0	8.0	6.0	0.0	7.0	6.0	3.0	6.0	3.0	10.0	14.0		
100	100	2.0	8.0	7.0	4.0	1.0	6.0	5.0	6.0	4.0	3.0	13.0	10.0		
100	500	5.0	5.0	9.0	7.0	0.0	2.0	2.0	4.0	1.0	3.0	6.0	9.0		
1000	20	5.0	5.2	8.7	3.2	6.3	5.9	6.7	4.9	4.4	5.7	10.2	7.1	5.7	4.7
1000	50	4.4	4.3	9.1	5.1	5.8	4.8	4.2	4.9	4.0	4.3	7.3	7.1		
1000	100	4.9	5.2	10.9	4.9	5.3	5.9	4.9	4.4	3.6	5.1	9.0	7.4		
1000	500	5.7	4.9	7.2	4.8	4.6	4.1	3.7	6.0	4.3	5.4	8.5	8.0		

Table 3 continued

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE- pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Initialization for tests with 300,000 function calls															
2	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0
2	50	0.0	0.0	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0		
2	100	0.0	0.0	0.0	50.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0		
2	500	0.0	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0	0.0	0.0		
10	20	10.0	0.0	30.0	0.0	10.0	10.0	10.0	10.0	20.0	0.0	30.0	30.0	10.0	20.0
10	50	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0		
10	100	0.0	10.0	0.0	0.0	0.0	20.0	10.0	0.0	0.0	0.0	0.0	10.0		
10	500	0.0	20.0	0.0	0.0	0.0	0.0	0.0	0.0	20.0	0.0	20.0	10.0		
30	20	6.7	3.3	6.7	3.3	6.7	3.3	3.3	3.3	6.7	6.7	10.0	10.0	6.7	6.7
30	50	0.0	0.0	3.3	10.0	0.0	6.7	0.0	0.0	0.0	10.0	6.7	0.0		
30	100	3.3	0.0	3.3	6.7	3.3	0.0	0.0	3.3	10.0	6.7	6.7	0.0		
30	500	3.3	3.3	20.0	3.3	3.3	3.3	10.0	3.3	3.3	3.3	6.7	0.0		
100	20	7.0	2.0	3.0	3.0	7.0	4.0	1.0	6.0	5.0	3.0	7.0	12.0	3.0	9.0
100	50	5.0	4.0	8.0	3.0	5.0	9.0	6.0	2.0	4.0	8.0	4.0	11.0		
100	100	5.0	6.0	4.0	3.0	5.0	4.0	0.0	6.0	8.0	5.0	6.0	13.0		
100	500	6.0	3.0	3.0	6.0	6.0	6.0	5.0	6.0	4.0	2.0	6.0	9.0		
1000	20	6.1	5.6	7.2	4.8	6.1	4.9	5.4	4.6	4.6	6.0	6.1	6.9	5.0	4.3
1000	50	4.3	5.5	7.6	6.0	4.3	5.3	5.3	5.6	5.5	5.3	7.5	7.2		
1000	100	5.0	4.5	6.5	5.4	5.0	5.3	4.1	4.8	4.3	5.9	6.9	7.4		
1000	500	3.7	4.6	7.7	4.4	3.7	5.6	4.0	4.3	4.9	5.5	7.3	7.8		

Table 4 Lowest p value obtained from Kolmogorov–Smirnov test among D dimensions. The H0 hypothesis, tested for each of D dimensions separately, is that the positions of best individuals are from uniform distribution after the final generation, i.e. after 10,000 (upper part of the table) or 300,000 (lower part of the table) function calls. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE
10,000 function calls								
2	20	5.40E−03	1.30E−03	3.95E−04	9.87E−144	2.51E−10	3.19E−01	1.27E−01
2	50	1.36E−03	8.74E−03	2.43E−06	2.62E−93	2.08E−09	1.38E−01	2.10E−01
2	100	5.97E−04	5.88E−03	1.64E−04	1.57E−78	1.35E−09	3.31E−01	2.21E−01
2	500	7.04E−06	7.29E−03	1.10E−02	4.24E−45	2.30E−09	3.69E−01	3.41E−01
10	20	4.63E−03	4.03E−04	5.57E−05	4.12E−103	2.24E−09	8.41E−03	3.11E−02
10	50	7.21E−05	1.14E−02	2.34E−03	3.80E−83	3.26E−10	8.36E−02	2.31E−01
10	100	1.14E−05	4.52E−04	8.31E−04	7.63E−71	6.70E−14	1.57E−01	6.29E−02
10	500	8.21E−07	1.14E−03	1.56E−06	2.39E−51	1.04E−11	1.00E−01	3.61E−03
30	20	4.18E−05	1.20E−03	2.52E−04	6.15E−96	4.34E−13	1.61E−02	1.82E−02
30	50	4.31E−05	9.11E−04	6.03E−04	1.77E−79	4.24E−12	7.97E−03	2.24E−02
30	100	6.89E−07	4.44E−04	9.76E−03	3.98E−67	2.04E−13	2.66E−02	1.96E−03
30	500	8.62E−08	3.82E−04	5.83E−03	1.64E−38	1.63E−13	6.97E−02	1.95E−02
100	20	3.67E−05	1.05E−03	5.30E−04	7.45E−96	2.77E−12	6.24E−03	8.25E−04
100	50	3.66E−07	1.89E−05	3.53E−04	3.15E−75	9.89E−15	8.94E−03	1.31E−03
100	100	6.97E−08	1.04E−06	4.36E−03	6.12E−77	1.14E−13	1.04E−02	9.64E−03
100	500	1.18E−08	1.13E−05	2.65E−04	7.65E−42	6.32E−15	1.96E−02	1.71E−02
1000	20	5.93E−06	2.64E−05	6.32E−06	1.58E−98	3.66E−14	2.71E−04	4.27E−04
1000	50	4.71E−08	1.07E−04	1.84E−04	2.46E−92	1.59E−15	2.62E−04	2.28E−04
1000	100	3.27E−09	2.53E−06	9.63E−05	2.30E−73	4.22E−17	1.54E−04	8.53E−04
1000	500	5.91E−10	2.16E−06	5.39E−05	3.96E−46	1.27E−16	2.94E−03	4.88E−04

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
10,000 function calls								
2	20	4.45E−02	3.40E−02	5.92E−02	4.11E−05	3.31E−06	3.02E−01	4.64E−07
2	50	2.46E−01	7.73E−03	8.73E−02	1.00E−04	7.82E−03		
2	100	5.45E−02	8.22E−03	2.10E−01	1.25E−06	4.65E−04		
2	500	1.45E−01	1.17E−03	2.14E−01	8.34E−05	3.83E−04		
10	20	5.58E−03	7.65E−04	5.82E−03	1.06E−06	3.06E−06	1.67E−02	1.07E−05
10	50	1.62E−02	7.83E−03	3.43E−03	4.72E−08	7.54E−05		
10	100	1.20E−03	2.20E−03	4.33E−03	3.17E−05	2.94E−04		
10	500	5.48E−05	4.18E−04	3.93E−02	1.21E−04	2.24E−03		
30	20	1.88E−03	3.13E−04	6.11E−04	1.26E−06	5.14E−06	6.18E−02	3.79E−03
30	50	6.66E−04	7.42E−05	1.94E−04	9.68E−08	1.33E−05		
30	100	1.77E−04	2.22E−04	3.44E−04	3.31E−06	6.30E−07		
30	500	1.49E−03	8.77E−05	2.84E−03	3.03E−06	1.87E−05		
100	20	7.39E−05	7.29E−06	2.54E−04	1.43E−07	1.21E−06	7.45E−04	7.56E−04
100	50	3.73E−04	2.09E−06	1.32E−05	2.45E−06	2.54E−08		

Table 4 continued

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
100	100	1.13E−04	3.57E−05	6.68E−05	4.65E−08	5.61E−08		
100	500	3.76E−05	1.61E−05	2.15E−03	1.44E−06	2.35E−05		
1000	20	5.04E−05	1.13E−06	7.47E−05	2.46E−07	6.85E−08	5.50E−04	6.82E−04
1000	50	3.94E−07	1.69E−06	1.15E−05	1.49E−09	1.07E−06		
1000	100	3.58E−06	1.71E−05	3.08E−06	2.55E−07	2.92E−07		
1000	500	1.56E−05	7.08E−07	2.48E−06	6.28E−07	1.44E−05		
D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE
300,000 function calls								
2	20	2.71E−02	2.02E−07	4.51E−06	7.17E−199	5.54E−06	1.53E−01	6.29E−01
2	50	4.07E−02	3.10E−07	1.75E−06	4.35E−157	2.79E−06	1.15E−01	7.00E−03
2	100	9.79E−02	1.36E−05	1.25E−05	5.17E−146	2.97E−09	1.17E−01	5.44E−01
2	500	1.89E−05	2.54E−03	2.87E−07	1.44E−126	1.59E−10	2.21E−01	5.44E−01
10	20	1.95E−02	6.77E−05	8.58E−08	5.98E−159	1.33E−06	6.67E−02	2.34E−01
10	50	1.77E−05	2.51E−04	4.20E−05	3.29E−139	1.40E−08	7.64E−02	2.54E−02
10	100	3.96E−04	1.66E−05	1.42E−07	2.56E−127	3.14E−12	2.31E−02	6.19E−02
10	500	5.34E−05	1.96E−02	4.04E−04	7.02E−100	3.49E−13	2.82E−01	2.89E−02
30	20	4.84E−03	9.32E−04	5.99E−04	1.51E−146	1.30E−08	1.16E−02	1.34E−03
30	50	8.15E−05	1.46E−05	1.28E−04	5.42E−119	1.17E−09	1.54E−03	3.09E−02
30	100	2.14E−03	9.44E−05	4.65E−04	3.68E−120	2.21E−10	2.05E−02	2.83E−02
30	500	5.28E−05	9.18E−06	9.16E−03	2.22E−90	9.87E−14	8.25E−03	1.61E−02
100	20	9.70E−04	3.26E−05	1.98E−04	3.33E−150	5.07E−10	2.42E−02	7.98E−04
100	50	1.41E−05	2.64E−04	1.91E−06	6.52E−136	4.96E−10	2.20E−02	3.12E−03
100	100	2.05E−04	1.53E−04	5.88E−05	2.21E−123	8.46E−13	1.85E−02	4.92E−05
100	500	2.59E−06	2.20E−04	1.18E−02	1.91E−93	4.74E−14	3.78E−03	1.73E−02
1000	20	6.54E−05	2.01E−05	1.43E−05	2.31E−157	1.22E−09	2.33E−03	5.00E−05
1000	50	3.70E−07	2.47E−06	6.59E−04	5.45E−144	1.10E−11	1.34E−03	3.15E−04
1000	100	7.96E−06	3.15E−05	5.51E−04	2.60E−126	3.43E−13	1.18E−03	3.41E−06
1000	500	5.14E−08	3.83E−05	1.76E−05	5.35E−102	1.24E−16	7.29E−04	1.71E−04
D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
300,000 function calls								
2	20	3.28E−01	3.74E−02	9.39E−03	3.24E−06	1.33E−04	2.28E−01	7.37E−07
2	50	5.86E−02	1.29E−01	1.86E−02	4.41E−05	4.04E−06		
2	100	6.68E−03	4.61E−01	2.75E−02	4.51E−05	4.32E−04		
2	500	1.17E−02	1.86E−01	4.36E−02	2.92E−06	2.37E−03		
10	20	2.15E−02	1.52E−03	5.63E−04	9.88E−06	1.15E−05	1.81E−02	2.78E−07
10	50	6.56E−03	1.26E−01	3.74E−04	9.50E−07	8.84E−04		
10	100	1.82E−02	4.50E−03	1.02E−03	1.76E−07	5.72E−04		
10	500	3.42E−02	1.48E−02	2.63E−03	1.11E−05	1.69E−05		
30	20	3.75E−03	1.58E−03	4.01E−04	7.06E−07	1.17E−06	2.76E−02	9.90E−05

Table 4 continued

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
30	50	1.68E−03	8.24E−05	7.79E−05	2.25E−08	6.22E−05		
30	100	1.02E−02	2.18E−03	3.62E−04	2.02E−05	1.15E−04		
30	500	7.08E−03	7.05E−03	1.58E−03	1.09E−07	1.77E−04		
100	20	1.42E−04	1.07E−03	7.14E−05	3.61E−07	2.56E−07	4.73E−03	1.72E−04
100	50	3.49E−04	4.49E−03	1.72E−05	1.86E−06	3.97E−07		
100	100	2.94E−03	6.00E−03	5.39E−06	6.43E−06	1.33E−05		
100	500	1.50E−03	3.04E−04	7.12E−05	4.68E−05	2.54E−06		
1000	20	2.59E−04	4.10E−05	1.06E−06	1.05E−07	1.13E−08	8.07E−06	4.98E−10
1000	50	4.92E−05	2.89E−04	8.48E−06	1.20E−08	2.50E−07		
1000	100	1.08E−03	1.06E−05	4.98E−06	6.33E−08	1.14E−06		
1000	500	2.04E−04	7.49E−05	2.11E−06	3.59E−07	1.48E−06		

Note that initialization is performed randomly and independently for each MNFC level. Therefore, the KS test produced different results after initialization for experiments in which algorithms were run with 10,000 versus 300,000 function calls (see Tables 3, 5). To show versatile effects of the structural bias on the location of the best solutions found by various algorithms for the fully random function f, edfs of 10-, 100- and 1000-dimensional variants of five algorithms (including two PSO, two DE and one GA variant) with *PopSize* set to 100 are illustrated graphically in Figs. 1–3 and in Supplementary Material available in an online version of the paper (Suppl. Fig. 1). In addition, four very different histograms of the p values obtained for the 1000-dimensional fully random function by four chosen algorithms (GA-MPC, DNS-PSO, CDE and DE-basic) with *PopSize* set to 100 are also illustrated in Fig. 4.

Tables 2 and 3 show that the percentage of H0 hypotheses rejected after the search had terminated is, for most algorithms, much higher than the 5% of the number of D values, indicating the presence of structural bias. This is the case for both 10,000 and 300,000 function calls. Hence, the presence of the structural bias affects the location of the best solutions found by most PSO and DE variants even after a relatively short run. More details on how quickly the effect of the structural bias may be observed are given in Sect. 4.2. The 14 tested algorithms may be roughly classified into six groups (G1–G6) according to the "strength" of the structural bias observed in Tables 2, 3, 4 and 5, which are discussed below.

Algorithms DNS-PSO and PSO-init-weight show very clear presence of structural bias (G1). In the case of both these algorithms, the KS test at $\alpha = 0.05$ (for all coordinates in all experiments, irrespective of the population size, problem dimensionality or maximum number of function calls) always rejects the H0 hypothesis that the empirical distribution of the best solutions found in 1000 runs is uniform. It indicates a very strong structural bias that may be attributed to the algorithm, as the distribution of values obtained during initialization is unbiased. The presence of extreme structural bias of DNS-PSO may be observed visually: the highly nonlinear edf for DNS-PSO (Fig. 1) and the narrow peak-like shape of the histogram for DNS-PSO p values (Fig. 4) confirm that the solutions are concentrated close to $x_j = 0$ on all dimensions ($j = 1, \ldots, 1000$), irrespective of the problem dimensionality.

In the case of three other algorithms (G2), namely PSO-basic, CLPSO-DEGL and GA-MPC, the H0 rejection rate is also very high. For CLPSO-DEGL and GA-MPC, the rejection

Table 5 Lowest p value obtained from Kolmogorov–Smirnov test among D dimensions. The H0 hypothesis, tested for each of D dimensions separately, is that the positions of best individuals are from uniform distribution after initialization. As initialization for tests with 10,000 and 300,000 function calls is performed independently, results for both cases are presented separately. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE
Initialization for tests with 10,000 function calls								
2	20	2.25E−01	6.23E−01	5.62E−02	7.81E−01	5.34E−01	7.26E−01	3.08E−01
2	50	4.47E−01	8.54E−01	1.16E−03	1.56E−01	2.49E−03	3.75E−01	2.93E−02
2	100	1.43E−01	4.08E−01	1.00E−01	2.15E−01	1.38E−01	6.32E−01	5.57E−01
2	500	5.38E−01	4.75E−01	2.72E−01	1.47E−01	3.78E−01	4.39E−01	1.50E−02
10	20	1.43E−01	1.66E−01	6.42E−02	3.96E−02	6.82E−03	1.20E−02	1.78E−01
10	50	6.65E−02	1.25E−01	5.08E−02	3.07E−01	1.49E−01	1.34E−02	1.11E−01
10	100	4.60E−02	1.13E−01	1.29E−02	3.26E−02	6.66E−02	9.11E−02	1.02E−01
10	500	5.61E−02	6.55E−02	6.92E−04	1.30E−01	1.03E−01	1.88E−02	3.28E−01
30	20	2.58E−02	2.50E−02	1.61E−02	2.49E−02	6.00E−02	1.00E−01	2.23E−02
30	50	7.71E−02	2.39E−02	7.87E−02	5.65E−02	3.60E−02	2.14E−03	7.90E−02
30	100	1.15E−02	7.21E−02	1.06E−02	1.65E−02	3.26E−02	3.78E−02	1.03E−01
30	500	4.43E−03	3.72E−02	2.19E−03	6.32E−02	1.60E−02	1.75E−03	4.60E−02
100	20	3.67E−03	1.29E−02	3.19E−03	1.15E−02	2.88E−04	1.90E−03	1.27E−02
100	50	2.08E−03	4.91E−03	4.80E−04	4.11E−03	6.30E−02	7.56E−04	2.84E−03
100	100	2.85E−02	2.01E−02	4.18E−03	2.51E−02	1.74E−02	9.31E−03	1.41E−02
100	500	6.52E−03	3.06E−03	1.48E−03	9.62E−03	6.41E−02	4.37E−03	2.51E−02
1000	20	3.39E−04	2.11E−04	4.51E−04	3.36E−03	4.60E−04	1.75E−03	4.31E−05
1000	50	3.13E−04	1.21E−03	8.58E−04	1.44E−03	6.91E−04	2.01E−04	2.24E−03
1000	100	1.80E−03	6.75E−04	2.61E−05	1.53E−03	4.76E−05	2.30E−04	1.93E−03
1000	500	1.58E−04	6.50E−04	9.75E−04	3.28E−04	4.15E−04	7.17E−04	1.15E−04

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Initialization for tests with 10,000 function calls								
2	20	3.23E−01	1.37E−01	3.19E−01	4.04E−01	6.18E−01	8.03E−01	3.59E−02
2	50	4.68E−01	4.57E−01	5.11E−02	9.44E−02	3.16E−01		
2	100	3.47E−01	1.80E−01	5.02E−01	3.50E−01	2.17E−01		
2	500	7.86E−02	4.77E−01	7.52E−01	9.38E−02	8.37E−02		
10	20	4.81E−02	9.57E−02	2.47E−01	3.54E−03	3.31E−02	7.13E−02	5.10E−02
10	50	4.51E−02	2.92E−02	8.77E−03	1.20E−01	3.29E−02		
10	100	1.24E−01	1.18E−01	5.99E−02	3.20E−02	8.25E−02		
10	500	1.00E−01	6.32E−02	2.03E−01	3.27E−04	2.23E−03		
30	20	2.31E−02	9.12E−03	1.24E−01	5.74E−02	5.31E−04	4.80E−02	1.38E−02
30	50	1.03E−02	6.14E−02	6.93E−02	2.71E−02	1.51E−02		
30	100	1.75E−03	5.59E−02	4.14E−02	3.03E−02	2.51E−03		
30	500	7.76E−02	2.19E−02	6.12E−02	2.77E−03	1.61E−03		
100	20	4.21E−03	1.63E−03	9.95E−03	1.51E−03	6.33E−04	5.72E−03	2.26E−02
100	50	6.80E−03	8.77E−03	1.25E−02	1.84E−03	4.04E−03		

Table 5 continued

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
100	100	1.79E−03	6.59E−03	1.14E−02	1.25E−04	3.82E−04		
100	500	3.63E−03	1.40E−02	3.06E−03	5.58E−04	7.33E−04		
1000	20	5.99E−05	2.91E−03	1.13E−04	4.83E−05	3.79E−04	2.77E−04	1.02E−03
1000	50	2.71E−04	3.34E−03	9.80E−05	5.42E−04	5.70E−04		
1000	100	2.26E−04	1.14E−03	5.30E−04	7.28E−05	1.16E−03		
1000	500	2.40E−05	6.55E−04	6.51E−04	1.18E−04	3.27E−05		
D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE
Initialization for tests with 300,000 function calls								
2	20	8.18E−01	3.26E−01	8.35E−01	2.81E−01	8.18E−01	5.85E−01	5.70E−01
2	50	2.03E−01	8.17E−01	3.94E−01	4.25E−01	2.03E−01	7.58E−01	3.65E−02
2	100	5.71E−01	2.66E−01	6.46E−01	2.15E−03	5.71E−01	1.53E−01	4.61E−01
2	500	2.14E−01	3.32E−02	2.38E−01	2.29E−01	2.14E−01	4.76E−01	4.61E−01
10	20	4.41E−02	6.47E−02	6.91E−03	8.34E−02	4.41E−02	3.33E−02	4.03E−02
10	50	1.75E−01	4.32E−02	1.94E−01	2.45E−01	1.75E−01	6.97E−02	1.22E−01
10	100	1.36E−01	3.94E−02	1.20E−01	9.33E−02	1.36E−01	1.15E−02	3.20E−03
10	500	1.78E−01	1.83E−02	1.58E−01	5.52E−02	1.78E−01	6.99E−02	6.58E−02
30	20	1.84E−02	1.70E−02	3.07E−02	3.81E−02	1.84E−02	3.53E−02	3.78E−02
30	50	7.98E−02	1.98E−01	4.80E−02	5.13E−03	7.98E−02	1.53E−02	7.39E−02
30	100	2.92E−02	8.03E−02	3.36E−02	1.23E−02	2.92E−02	5.02E−02	5.08E−02
30	500	4.91E−02	3.42E−02	9.29E−03	1.75E−04	4.91E−02	1.57E−02	1.71E−02
100	20	1.10E−02	2.41E−03	1.57E−02	2.84E−03	1.10E−02	5.27E−03	3.97E−03
100	50	2.98E−03	2.04E−03	8.32E−03	2.21E−03	2.98E−03	3.45E−03	2.20E−03
100	100	1.32E−02	6.55E−03	5.17E−03	8.92E−03	1.32E−02	1.53E−02	6.93E−02
100	500	2.02E−02	6.28E−03	4.12E−03	1.39E−03	2.02E−02	3.37E−03	3.44E−03
1000	20	8.27E−04	1.17E−03	1.77E−04	1.23E−04	8.27E−04	1.55E−04	2.96E−06
1000	50	4.45E−03	4.65E−04	1.74E−03	1.63E−03	4.45E−03	1.56E−03	1.54E−04
1000	100	1.56E−04	2.63E−04	9.25E−04	2.55E−03	1.56E−04	8.96E−04	1.45E−03
1000	500	5.00E−04	3.71E−04	9.56E−04	2.46E−03	5.00E−04	8.47E−04	7.42E−04
D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Initialization for tests with 300,000 function calls								
2	20	2.49E−01	1.68E−01	4.12E−01	3.00E−02	5.79E−01	1.15E−01	1.18E−01
2	50	1.20E−01	4.71E−01	3.44E−01	4.21E−01	3.55E−01		
2	100	1.51E−02	4.59E−01	8.41E−01	1.14E−01	5.87E−01		
2	500	6.36E−01	8.55E−01	2.60E−02	4.03E−01	2.69E−01		
10	20	4.28E−02	1.89E−02	5.33E−02	2.72E−02	1.78E−03	3.05E−02	2.06E−02
10	50	1.61E−01	1.52E−01	1.13E−02	6.92E−02	1.26E−01		
10	100	1.34E−01	7.21E−02	1.16E−01	9.07E−02	4.70E−02		
10	500	2.66E−01	8.54E−03	9.86E−02	3.00E−03	4.61E−02		
30	20	3.00E−02	4.61E−02	2.11E−02	4.39E−03	2.42E−02	2.44E−02	6.10E−03

Table 5 continued

D	PopSize	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
30	50	6.83E−02	1.38E−01	2.22E−02	1.38E−02	5.39E−02		
30	100	3.02E−02	2.49E−02	2.88E−02	1.27E−02	6.77E−02		
30	500	2.95E−02	1.52E−02	4.97E−02	1.97E−02	1.16E−01		
100	20	9.50E−03	2.20E−02	5.32E−03	2.49E−03	6.04E−04	1.64E−02	6.14E−03
100	50	3.79E−03	2.45E−02	1.18E−03	6.75E−03	3.30E−03		
100	100	5.76E−03	1.43E−02	2.96E−02	1.81E−03	2.61E−03		
100	500	6.91E−03	1.24E−02	2.65E−02	1.17E−02	1.21E−04		
1000	20	1.79E−04	3.40E−03	1.25E−03	1.47E−03	4.29E−04	1.83E−03	1.87E−03
1000	50	4.34E−04	2.82E−04	1.85E−04	3.99E−05	5.49E−05		
1000	100	5.83E−04	5.56E−04	1.15E−04	1.25E−03	1.07E−04		
1000	500	1.09E−03	3.54E−04	2.64E−03	2.02E−04	1.36E−03		

rate is predominantly within the 75–100% range, with an average exceeding 90%. For PSO-basic, the rejection rate is slightly lower. Irrespective of the problem dimensionality, in the case of the PSO-basic algorithm, the percentage of rejected H0 hypotheses increases with the population size, often reaching 90–100% for the largest population sizes tested (see Table 2). Interestingly, the opposite relation between the population size and the percentage of H0 rejection rates is observed for CLPSO-DEGL and GA-MPC. As illustrated in Fig. 2 (right side), in the case of GA-MPC, the edf is mildly S-shaped indicating less evident structural bias. Contrary to DNS-PSO, the best solutions concentrate rather in the middle of the search space, and the divergence from the uniform distribution is much weaker. However, a selected histogram of p values obtained for GA-MPC, given in Fig. 4, shows that the distribution of p values is highly skewed. What is important is that the distribution of the best solutions obtained after initialization can again be considered uniform (see Tables 3, 5).

In the case of four other metaheuristics (ALC-PSO, DEGL, MDE_pBX and SADE), the presence of structural bias is also quite clearly observed (G3). However, although for these algorithms the H0 rejection rates are much higher than 5%, they are not as high as in the case of the five algorithms discussed earlier. Also, H0 rejection rates vary depending on the problem dimensionality, the population size and MNFC. Except for SADE which seems to be the most biased within this group, the H0 rejection rates are well below 50% for the vast majority of higher-dimensional experiments. In the case of DEGL and MDE_pBX, H0 rejection rates are often within the range of 10–30%. Generally, the structural bias observed seems the weakest when the population size of SADE, MDE_pBX and ALC-PSO is set to the highest tested value (500 particles or individuals). However, in the case of the four considered algorithms the relation between the strength of structural bias and the population size does not show clear trends and hence cannot be generalized. For example, for the DEGL algorithm, the percentage of rejected H0 hypotheses is often the lowest when the population size is the highest. In the case of ALC-PSO, the structural bias is the strongest for low-dimensional problems. When the population size is lower than 500, and the number of function calls is set to 300,000, the structural bias of ALC-PSO is observed for all dimensions of 2- and 10-dimensional experiments. Hence, due to the fact that a large population size is never used in practice for the PSO algorithms, de facto ALC-PSO suffers from strong structural bias.

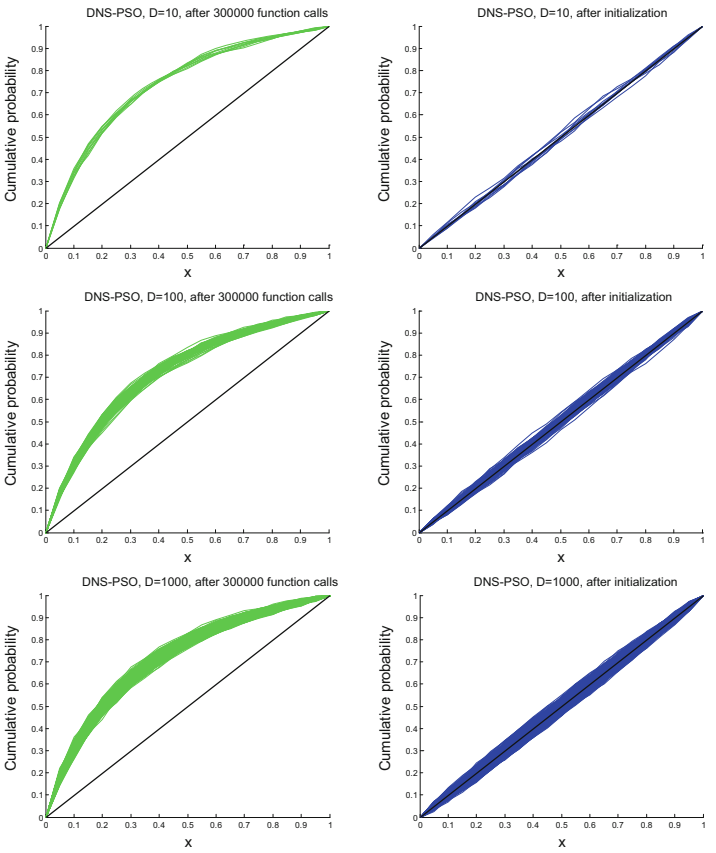

Fig. 1 Empirical distribution function (edf) of the localization of the best solutions found by DNS-PSO algorithm with population size set to 100 individuals. Left figures show edf of best solutions found after 300,000 function calls (*green lines*); the right figures show edf of best solutions found after random initialization (*blue lines*). Each *green line* and each *blue line* represent edf for a single axis out of D (D varies from 10 in upper figures, to 1000 in the lower figures) (Color figure online)

Here again, the distributions of best solutions obtained after initialization indicate the lack of structural bias.

It is worth noting that for many of the algorithms included into groups G2 and G3, almost the same, S-shaped edf of best solutions found after 300,000 function calls is observed—three chosen examples (ALC-PSO, SADE and GA-MPC) are illustrated in Fig 2. Again, the edf of best solutions obtained after initialization of all these algorithms does not show any signs of structural bias (see Suppl. Fig. 1). As such examples include both DE, PSO and GA variants, this may suggest that metaheuristics from various families of methods may too frequently sample locations in the "middle" part of the search space. Interestingly, the shape of edf does not depend on the chosen dimensionality of f.

For two tested algorithms (G4), CLPSO and CDE, the presence of structural bias may be said to be disputable (see also Fig. 4 for an illustrative example of the histogram of p values of CDE). When CLPSO is applied to low-dimensional problems, the presence of structural bias is observed in 100% of cases. However, this number diminishes with increase in the problem dimensionality, and for 100- or 1000-dimensional problems, the percentage of H0 rejections

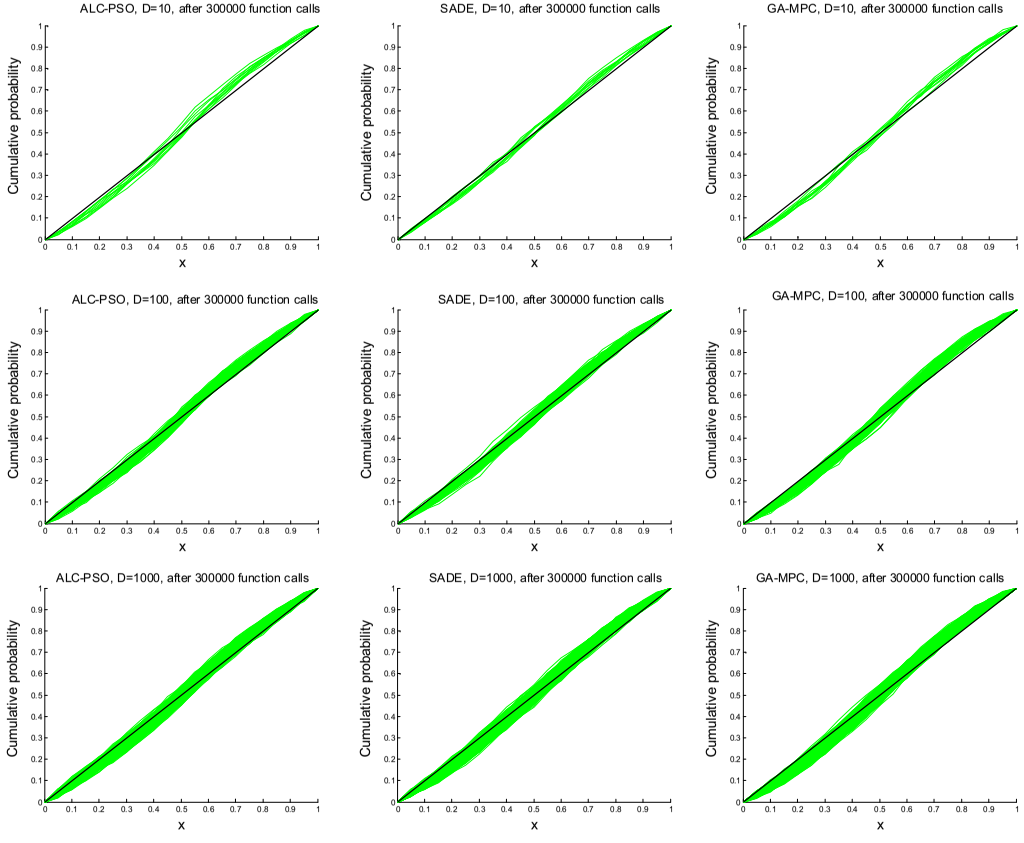

Fig. 2 Empirical distribution function (edf) of the localization of the best solutions found after 300,000 function calls by ALC-PSO (*left*), SADE (*middle*) and GA-MPC (*right column*) algorithms, when the population size is set to 100. Each *green line* represents edf for a single axis out of D (D varies from 10 in upper figures, to 1000 in the lower figures). The respective edfs of the best solutions found in the initialized populations, that show no sign of structural bias, are given in Supplementary Material available online (Suppl. Fig. 1) (Color figure online)

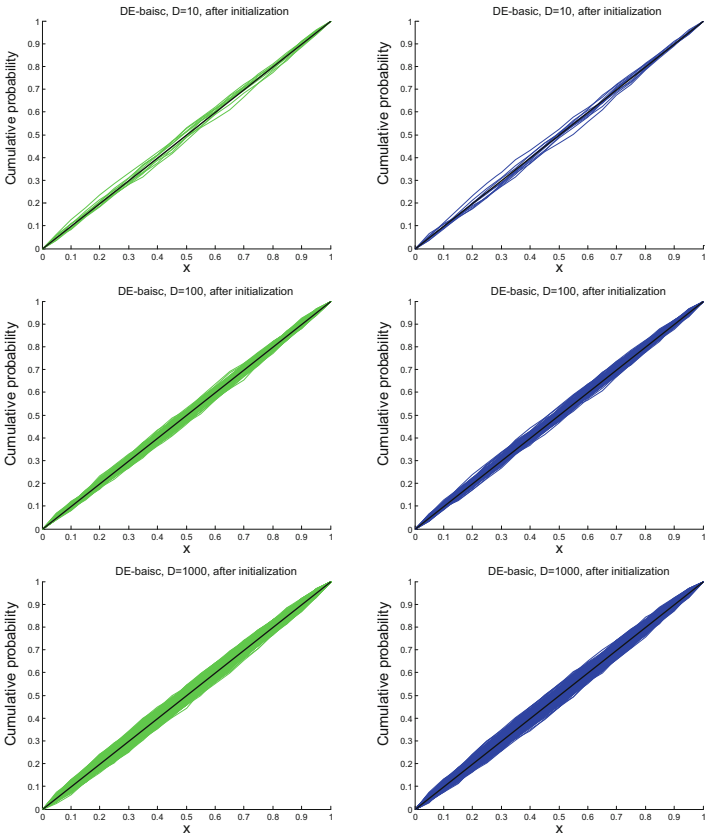

Fig. 3 Empirical distribution function (edf) of the localization of the best solutions found by DE-basic algorithm with population size set to 100 individuals. Left figures show edf of best solutions found after 300,000 function calls (*green lines*); the right figures show edf of best solutions found after random initialization (*blue lines*). Each *green line* and each *blue line* represent edf for a single axis out of D (D varies from 10 in upper figures, to 1000 in the lower figures) (Color figure online)

for CLPSO varies between 7 and 25%. This shows a relatively weak bias. The strength of structural bias is similar for 10,000 and 300,000 function calls, and the relationship between the number of H0 rejections and the population size is not straightforward. However, there may be some questions regarding the distribution of solutions obtained after CLPSO pseudo-random initialization. In the case of the CLPSO algorithm (and also CLPSO-DEGL, which uses the same CLPSO code), the number of rejected H0 hypotheses for each experiment after initialization remains between 5 and 10%. This is a higher level than the expected 5% and higher than observed for other algorithms. As indicated in Table 1, the CLPSO source code has been obtained from one of the authors of Liang et al. (2006). We consider this code as the official version of the CLPSO algorithm and use it "as is", especially since it has also been applied by many other users. For higher-dimensional problems, the number of H0 hypotheses rejected for distributions of the best solutions obtained after termination of the algorithm is only slightly larger than for the initialized ones. It implies that the structural bias of CLPSO is either non-existent or very weak when problem dimensionality is high.

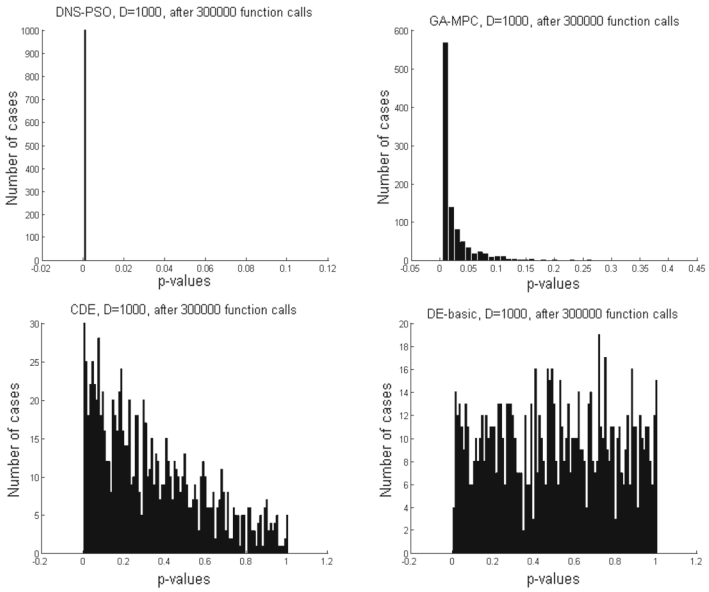

Fig. 4 Histogram of p values obtained for four selected algorithms with population size set to 100 individuals and MNFC set to 300,000

Hence, the supposed structural bias of CLPSO is observed for low-dimensional problems, but becomes barely detectable for high-dimensional ones.

In the case of CDE, the opposite relationship between the presence of structural bias and the problem dimensionality is noted—the CDE algorithm seems to be unbiased for low-dimensional problems. For higher-dimensional problems, the presence of structural bias in CDE depends on the population size: the higher the population size, the weaker the structural bias. When the number of individuals is set to 500 [which, contrary to PSO methods, is not an unreasonable choice for DE algorithms, especially when the problem dimensionality is high; see Storn and Price (1997), Gämperle et al. (2002) and Piotrowski (2016)], no presence of structural bias is noted even for high-dimensional problems. However, the CDE algorithm is biased for lower population sizes. This is illustrated by the histogram of p values for the 1000-dimensional case with $PopSize = 100$ and MNFC set to 300,000 (Fig. 4). Nonetheless, the structural bias is relatively "weak" compared to most other methods tested. As a result, CDE is the only tested metaheuristic among those proposed in the present century, which may be considered structurally unbiased, at least when it is used with a large but still reasonable population size.

We can also conclude, based on Tables 2, 3, 4 and 5, that the legendary Nelder–Mead algorithm (NMA) invented in 1965 and the basic DE variant (DE-basic) proposed in 1995 are structurally unbiased (G5). This result is so clear that it does not require lengthy discussion. The shapes of the edfs of the best solutions found by DE-basic are linear, irrespective of the dimensionality (see Fig 3). In further support of our findings, we provide a histogram of p values for the DE-basic algorithm for the 1000-dimensional fully random function, with $PopSize$ set to 100 and MNFC $= 300{,}000$ (see Fig. 4). Although each of the four histograms presented in Fig. 4 clearly differs from the others, the only one that does not point at any bias in the results is obtained from the DE-basic algorithm.

At first glance it may seem that RA, another 50-year-old approach, is also structurally unbiased. However, the tests performed on functions like f seem rather inconclusive for RA. This explains why we put it into a separate group (G6) for which structural bias cannot be reliably measured by the approach discussed. When testing any algorithm on f (Eq. (1)), the expected value of success (finding better solution than all sampled previously in that run) in the nth function call ($n = 1, \ldots,$ MNFC) equals $E\,[Pr(f(n) < \min(f(n-1), f(n-2), \ldots, f(1))] = \frac{1}{n}$, where Pr stands for probability. RA is not a rotation invariant method, and it makes moves along coordinate axes, in each run starting from the same coordinate system. It is only under special circumstances that it may rotate its coordinate system during run, namely after one successful and one unsuccessful step in each direction. The objective function is called for the first time ($n = 1$) by the initial solution, hence this initial solution must be successful ($Pr = 1$), the first move along the first coordinate leads to the second function call ($n = 2$) with the expected probability of success $E\,[Pr(f(2) < f(1))] = \frac{1}{2}$ and so on. When moves along all D coordinate axes are tried, the loop restarts. Let us denote the number of loops already completed as nl. When the loop restarts, $n = nl \cdot D + 2$ for move along the first coordinate axe, for the move along the second coordinate axe $n = nl \cdot D + 3$, etc. The later the move along a particular coordinate is executed, the lower the probability that such move will be successful on f defined by Eq. (1). Also, the more loops completed, the lower the probability that any move within any subsequent loop in a sequence will be successful. To change the location of the best solution generated during initialization along particular coordinate axe, either the move along such a coordinate must be successful (which is rare after the first loop), or the coordinate system must be rotated to allow for any move in the new coordinate system to be successful. However, in RA, the rotation of coordinates may take place only after one successful and one unsuccessful step in each direction. This, as shown above, is unrealistic for higher-dimensional variants of f. As a result, along many coordinates no successful moves occur in high-dimensional cases and, consequently, no algorithm-induced structural bias can be observed. In the case of low-dimensional problems, successful moves may occur for all coordinates and the presence of structural bias may be (and indeed is) observed. As a result, the experiment with RA seems useful not as far as detecting the structural bias is concerned, but to show some limitations of the concept.

The list of the lowest p values presented in Tables 4 and 5 supports the above conclusions. However, knowing the lowest p values may allow us to spot differences in the strength of structural bias. Based on the number of H0 rejection rates at $\alpha = 0.05$, both DNS-PSO and PSO-init-weight could be considered equally biased. But comparison of the lowest p values from Tables 4 and 5 reveals that they differ by dozens of orders of magnitude. For example, the lowest noted p value obtained when PSO-init-weight is run for 300,000 function calls on the 1000-dimensional function f with population size set to 50 (which is a frequent choice for the PSO algorithms) equals 1.10e-11, but the respective p value for DNS-PSO is as low as… 5.45e-144. Also histograms of p values obtained for four algorithms (GA-MPC, DNS-PSO, CDE and DE-basic) on the 1000-dimensional fully random function with population size set to 100 shown in Fig. 4 imply large differences between different algorithms in terms of the strength of structural bias.

When the two-sample KS test is used to verify the hypothesis if the initial and the final edfs of a particular algorithm differ, the results generally confirm conclusions based on the one-sample KS test (see Tables 6, 7 for numbers of rejected H0 hypotheses when the maximum number of function calls is set to 10,000 and 300,000, respectively). This is not surprising, as we found out that the initial sampling could be considered unbiased, based on results of testing the hypothesis that the distributions of the best solutions after initialization of each

Table 6 Percentages of H0 hypotheses rejected by two-sample Kolmogorov–Smirnov test at $\alpha = 0.05$ significance level. The H0 hypothesis is that the positions of best individuals among those generated during initialization from uniform distribution, and of best individuals obtained after 10,000 function calls by particular algorithm are from the same distribution. H0 is tested in each among D dimensions separately. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
2	20	100.0	100.0	50.0	100.0	100.0	0.0	0.0	50.0	50.0	0.0	100.0	50.0	0.0	50.0
2	50	100.0	0.0	100.0	100.0	100.0	0.0	50.0	0.0	50.0	50.0	0.0	50.0		
2	100	100.0	50.0	100.0	100.0	100.0	0.0	0.0	0.0	100.0	0.0	100.0	100.0		
2	500	100.0	50.0	0.0	100.0	100.0	0.0	0.0	50.0	50.0	0.0	100.0	0.0		
10	20	30.0	20.0	30.0	100.0	100.0	20.0	10.0	40.0	50.0	0.0	80.0	60.0	10.0	10.0
10	50	40.0	20.0	30.0	100.0	100.0	0.0	0.0	20.0	30.0	40.0	100.0	50.0		
10	100	100.0	30.0	50.0	100.0	100.0	0.0	0.0	30.0	40.0	20.0	80.0	70.0		
10	500	70.0	30.0	70.0	100.0	100.0	10.0	10.0	30.0	60.0	0.0	50.0	50.0		
30	20	33.3	13.3	13.3	100.0	100.0	0.0	13.3	6.7	23.3	26.7	93.3	86.7	3.3	3.3
30	50	60.0	20.0	16.7	100.0	100.0	6.7	6.7	23.3	16.7	23.3	83.3	73.3		
30	100	76.7	26.7	6.7	100.0	100.0	0.0	6.7	36.7	36.7	13.3	80.0	60.0		
30	500	100.0	16.7	10.0	100.0	100.0	6.7	0.0	10.0	53.3	13.3	60.0	36.7		
100	20	43.0	15.0	10.0	100.0	100.0	8.0	6.0	21.0	35.0	29.0	82.0	66.0	7.0	1.0
100	50	65.0	18.0	14.0	100.0	100.0	5.0	9.0	26.0	29.0	29.0	68.0	60.0		
100	100	66.0	28.0	10.0	100.0	100.0	9.0	3.0	29.0	33.0	18.0	74.0	38.0		
100	500	89.0	34.0	8.0	100.0	100.0	5.0	5.0	20.0	47.0	12.0	47.0	46.0		
1000	20	33.8	11.4	7.6	100.0	100.0	5.7	8.9	20.0	40.5	27.4	66.0	76.4	5.9	0.0
1000	50	57.0	14.8	8.0	100.0	100.0	4.4	5.5	24.1	26.8	26.1	73.9	56.1		
1000	100	77.3	32.5	9.9	100.0	100.0	5.2	5.1	20.6	26.3	18.1	66.5	44.8		
1000	500	88.1	33.1	8.5	100.0	100.0	4.1	5.3	21.5	49.1	14.5	49.0	34.8		

Table 7 Percentages of H0 hypotheses rejected by two-sample Kolmogorov–Smirnov test at $\alpha = 0.05$ significance level. The H0 hypothesis is that the positions of best individuals among those generated during initialization from uniform distribution, and of best individuals obtained after 300,000 function calls by particular algorithm are from the same distribution. H0 is tested in each among D dimensions separately. NMA has always population size equal $D + 1$; RA is not population based

D	PopSize	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
2	20	0.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	50.0
2	50	0.0	100.0	100.0	100.0	100.0	0.0	0.0	50.0	0.0	50.0	100.0	100.0		
2	100	0.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	50.0	100.0		
2	500	50.0	0.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	50.0	100.0	100.0		
10	20	20.0	60.0	70.0	100.0	100.0	0.0	10.0	0.0	20.0	40.0	100.0	80.0	0.0	10.0
10	50	20.0	60.0	30.0	100.0	100.0	0.0	10.0	30.0	0.0	40.0	100.0	60.0		
10	100	20.0	70.0	50.0	100.0	100.0	20.0	0.0	0.0	0.0	10.0	80.0	60.0		
10	500	40.0	30.0	30.0	100.0	100.0	0.0	0.0	0.0	30.0	30.0	80.0	50.0		
30	20	6.7	20.0	20.0	100.0	96.7	3.3	10.0	16.7	13.3	43.3	93.3	90.0	3.3	0.0
30	50	10.0	40.0	33.3	100.0	100.0	6.7	3.3	13.3	10.0	36.7	96.7	53.3		
30	100	30.0	26.7	26.7	100.0	100.0	6.7	6.7	3.3	10.0	20.0	83.3	50.0		
30	500	46.7	30.0	23.3	100.0	100.0	6.7	13.3	3.3	13.3	20.0	66.7	50.0		
100	20	8.0	27.0	18.0	100.0	94.0	2.0	15.0	5.0	11.0	30.0	76.0	78.0	5.0	1.0
100	50	29.0	32.0	12.0	100.0	99.0	2.0	5.0	18.0	14.0	37.0	66.0	63.0		
100	100	22.0	29.0	16.0	100.0	100.0	4.0	1.0	12.0	10.0	31.0	56.0	59.0		
100	500	50.0	11.0	7.0	100.0	100.0	5.0	5.0	9.0	12.0	26.0	45.0	50.0		
1000	20	10.2	21.3	6.4	100.0	93.1	4.9	13.7	7.6	16.8	37.2	63.1	85.1	3.7	0.1
1000	50	16.4	26.9	7.3	100.0	99.7	4.6	7.7	12.9	13.6	31.4	67.6	58.0		
1000	100	29.0	23.9	5.9	100.0	100.0	4.6	7.1	9.5	11.6	31.7	55.2	53.2		
1000	500	54.5	18.9	7.2	100.0	100.0	5.1	4.3	12.2	11.1	28.6	56.4	47.5		

algorithm are uniform (H0 rejection rates are close to 5%). The exceptions observed for CLPSO and CLPSO-DEGL were discussed earlier.

According to the above discussion, the structural bias is observed both in most PSO and in most DE variants tested. However, according to the results given in Tables 2, 3, 4, 5, 6 and 7, the PSO algorithms seem more structurally biased than the DE algorithms. This may be due to the fact that the basic DE version proposed by Storn and Price (1995) does not exhibit any sign of structural bias, but the basic PSO version proposed by Eberhart and Kennedy (1995) does. Only one PSO variant (CLPSO) may be considered almost structurally unbiased, at least in some experiments. In general, the number of H0 hypotheses rejected by the KS test is much higher for the PSO algorithms than for the DE algorithms. Nonetheless, apart from DE-basic and (in some cases) CDE, the presence of structural bias is also observed in the DE algorithms.

One can expect that the combination of two approaches, when one is structurally biased and the second is (almost) not, will likely be structurally biased. Although this observation is based on only one example, namely the CLPSO-DEGL hybrid (see Tables 2, 3, 4, 5), it is hard to explain how one constituent algorithm could correct the effect of the other's structural bias.

4.2 How quickly structural bias affects the search

The above discussion concluded that the effect of structural bias is observed within 10,000 function calls. Moreover, for most algorithms, the strength of structural bias seems to be similar after 10,000 and 300,000 function calls, and in the case of a few methods, it may even decrease. Because in all cases but CLPSO and CLPSO-DEGL the initial solutions are unbiased, the structural bias has to affect the early stages of the search. Here a natural question arises as to how quickly this bias impacts the distribution of the best solutions found. To address this question, additional tests with very low numbers of function calls (between 200 and 5000) were conducted by running each algorithm with *PopSize* set to 100 (apart from NMA and RA, which require population sizes set to $D + 1$ and 1 respectively) on a 1000-dimensional version of the fully random function f (Eq. (1)). The results are given in Table 8 which also includes the values obtained after initialization and 10,000 function calls copied from Tables 2 and 3. Note that, in the case of NMA and RA, some rows in Table 8 are empty. This is because NMA requires 2002 function calls (1001 for initialization and 1001 for the first run) to make a first move, and RA needs at least 1001 function calls (1 for initialization and 1000 to sample along each coordinate) to present any meaningful results. However, for simplicity, in Table 8 the first available results are given in the same row as the results obtained after 1000 (RA) and 2000 (NMA) function calls, but are marked with *.

It is very important to note here that the results shown in Table 8 for various numbers of function calls are mutually independent. Each time every algorithm was initialized independently. Consequently, results reported after 300 function calls come from other runs than those reported after 200 calls, and so on. This may lead to some occasional bumps in specific values, but avoids the problem of dependency between the values observed.

We can conclude from Table 8 that the structural bias, if present, affects the distribution of the best solutions very quickly. However, there are clear differences between the PSO and DE algorithms in this respect. For PSO, in the case of three out of five variants tested, the bias developed already after the first iteration (e.g. one move of each particle). Afterwards the number of H0 hypotheses rejected remains more or less constant in subsequent generations, although p values may further diminish the "strengthening" effect of the structural bias. The rapid effects of the structural bias present in DNS-PSO may be observed in Fig. 5. The edf of

Table 8 Percentages of H0 hypotheses rejected by Kolmogorov–Smirnov test at $\alpha = 0.05$ significance level after very few numbers of function calls.Tests are performed for 1000-dimensional fully random function, *PopSize* = 100 for all algorithms but NMA (which population always equal $D + 1$) and RA (which population is always set to 1). The H0 hypothesis is that the positions of best individuals after specified number of function calls are from the uniform distribution within [0, 1] search domain. H0 is tested in each among D dimensions separately. Tests for each number of function calls are independent, but results obtained after initialization and after 10,000 function calls are the same as given in Table 2 (results for initialization of other experiments are skipped). NMA requires at least 2002 function calls (1001 for initialization and 1001 for the first run); RA requires at least 1001 calls (1 for initialization and 1000 to sample along each dimensions), but for simplicity the respective first NMA and RA results are given in the same row as the results obtained after 2000 (NMA) or 1000 (RA) function calls

Number of function calls	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Initialization	4.9	5.2	10.9	4.9	5.3	5.9	4.9	4.4	3.6	5.1	9.0	7.4	5.7	4.7
200	99.1	68.1	7.2	72.2	99.8	6.0	4.5	15.8	14.3	6.2	27.4	28.8		
300	100.0	7.9	8.2	100.0	100.0	4.8	5.4	26.3	26.6	8.6	35.1	40.9		
400	91.2	19.5	9.0	96.3	100.0	3.6	6.6	31.6	43.0	11.4	49.2	39.5		
500	52.4	32.8	10.2	100.0	100.0	3.9	4.5	42.4	55.5	11.4	51.3	44.5		
1000	97.7	43.2	7.6	100.0	100.0	5.8	4.8	57.6	85.2	21.2	76.2	65.2		5.2
2000	99.8	59.6	7.3	100.0	100.0	4.2	6.3	56.7	90.4	26.7	88.6	78.2	5.9	5.3
5000	99.9	60.5	7.3	100.0	100.0	5.4	5.3	53.5	79.2	37.3	90.2	82.4	5.2	5.5
10,000	99.2	64.3	9.3	100.0	100.0	4.6	4.5	44.8	54.4	40.1	93.6	80.4	4.9	4.8

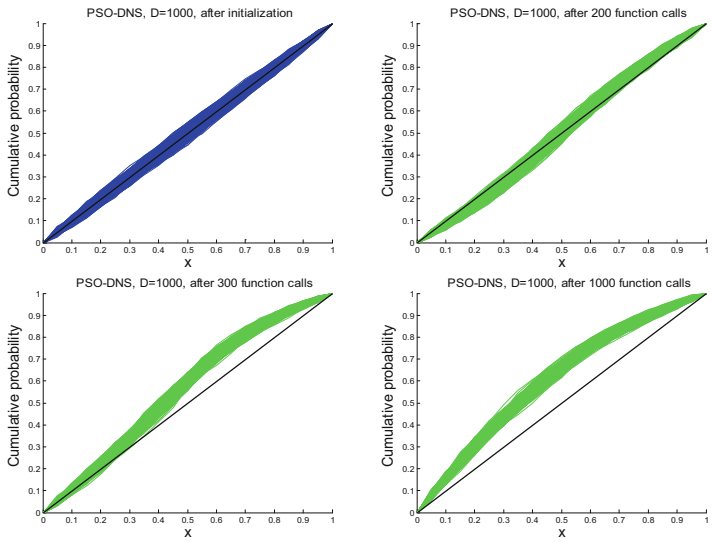

Fig. 5 Empirical distribution function (edf) of the localization of best solutions of 1000-dimensional fully random function found by DNS-PSO algorithm with population size set to 100 individuals after various numbers of function calls. Each *green line* and each *blue line* represent edf for a single axis out of D ($D = 1000$) (Color figure online)

the best solutions found for the initialized population (hence after 100 function calls) is linear, but just after 200 function calls the edf becomes S-shaped (as discussed earlier—this is a frequent shape observed for many metaheuristics). From 300 function calls, the edf becomes more and more bent-shaped, indicating that the majority of the best solutions found by the algorithm are located close to $\mathbf{x} = 0$.

On the other hand, in the case of DE algorithms and the GA variant tested, the effect of the structural bias is low after the first few generations. Then, the effect of structural bias often steadily increases during early generations and reaches a maximum after a few or several thousands of function calls. After that, the strength of the structural bias may even start receding (at least in the case of DEGL and MDE_pBX variants). The reason for this remains unclear and needs to be addressed in the future by a separate, algorithm-focused study. The slower development of the structural bias in the case of DE variants is illustrated in Fig. 6, using the MDE_pBX algorithm.

The results given in Table 8 agree with findings from Sect. 4.1 that some algorithms (NMA and DE-basic) are structurally unbiased, and a few others may be called almost structurally unbiased (CDE, CLPSO). The lack of changes in the shape of edf of the DE-basic algorithm with the increasing number of function calls is illustrated in Fig. 7, which visually confirms that there is no sign of structural bias in the distribution of best solutions found after any number of function calls when the classical DE scheme is used.

4.3 Can we identify reasons for the structural bias?

Searching for the parts of each tested algorithm that cause structural bias must unfortunately be considered beyond the scope of the present study. Such research probably needs to be performed for each specific approach separately, although we cannot exclude the possibility that the reasons for, for example, the frequently observed S-shaped edf may be similar

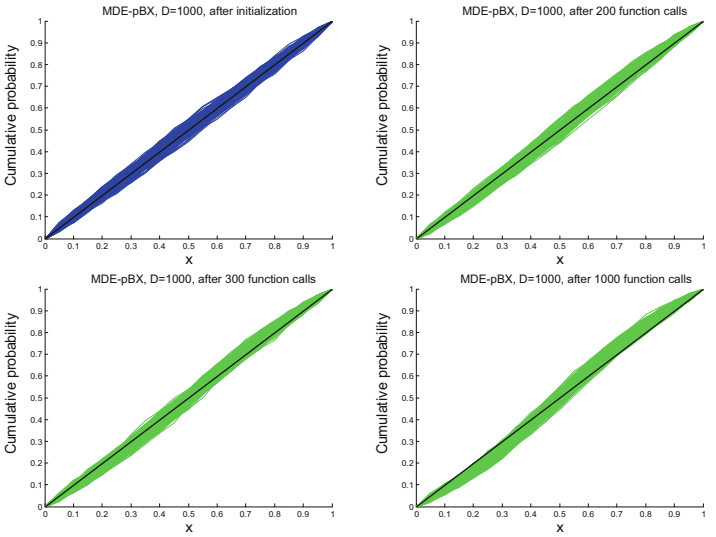

Fig. 6 Empirical distribution function (edf) of the localization of best solutions of 1000-dimensional fully random function found by MDE-pBX algorithm with population size set to 100 individuals after various numbers of function calls. Each *green line* and each *blue line* represent edf for a single axis out of D ($D = 1000$) (Color figure online)

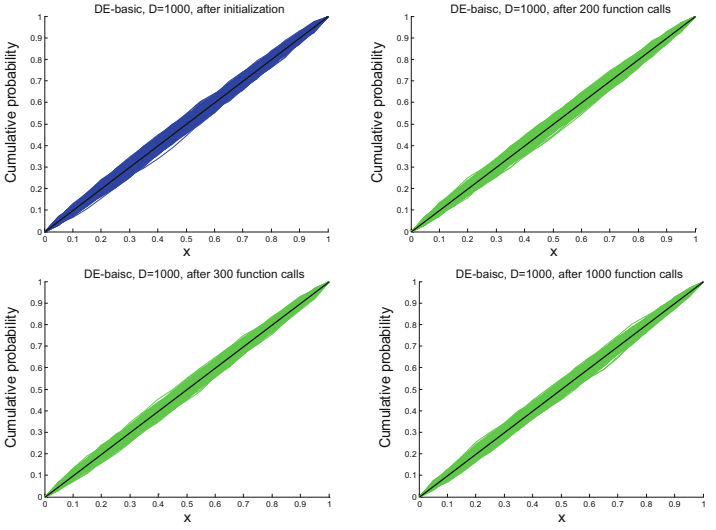

Fig. 7 Empirical distribution function (edf) of the localization of best solutions of 1000-dimensional fully random function found by DE-basic algorithm with population size set to 100 individuals after various numbers of function calls. Each *green line* and each *blue line* represent edf for a single axis out of D ($D = 1000$) (Color figure online)

for various metaheuristics. However, as an example for future studies it may be useful to investigate reasons for the presence of the extremely "strong" structural bias in the DNS-PSO algorithm that tends to locate best solutions close to $\mathbf{x} = 0$. In DNS-PSO (Wang et al. 2013), local and global neighbourhood search strategies are carried out, both aiming

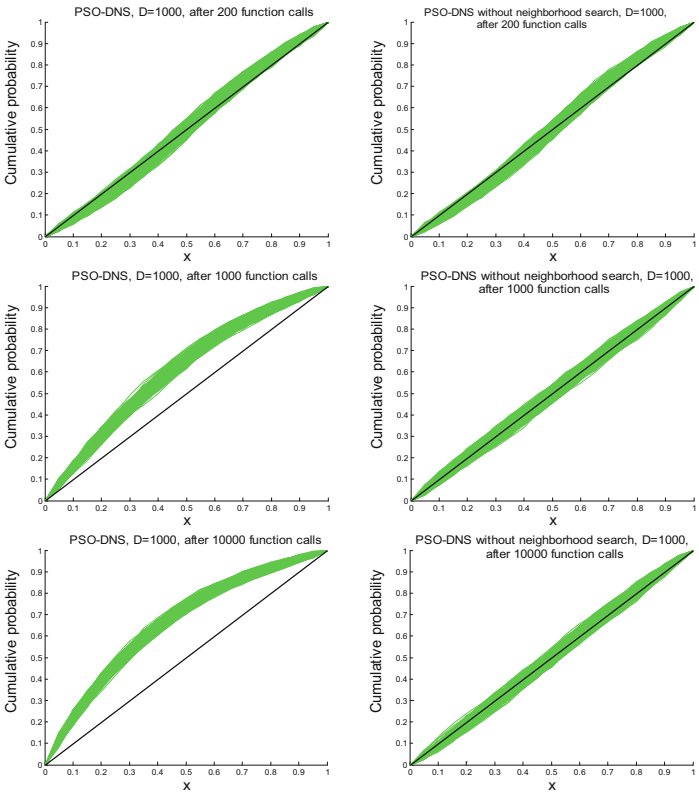

Fig. 8 Empirical distribution function (edf) of the localization of best solutions of 1000-dimensional fully random function found by DNS-PSO algorithm and version of DNS-PSO with neighbourhood strategies skipped, each with population size set to 100 individuals. Results obtained after 200, 1000 and 10,000 function calls (runs are independent) are shown. Each *green line* represents edf for a single axis out of D ($D = 1000$) (Color figure online)

at creation of additional temporary particles $\mathbf{LX}_i$ and $\mathbf{GX}_i$ for which the function value is sampled. These particles are generated as $\mathbf{LX}_i = r_1 \cdot \mathbf{X}_i + r_2 \cdot \mathbf{pbest}_i + r_3 \cdot (\mathbf{X}_c - \mathbf{X}_d)$ and $\mathbf{GX}_i = r_4 \cdot \mathbf{X}_i + r_5 \cdot \mathbf{pbest}_i + r_6 \cdot (\mathbf{X}_e - \mathbf{X}_f)$, where $\mathbf{X}_i$ is the current position vector of the ith particle, $\mathbf{pbest}_i$ is the best position sampled so far by particle $\mathbf{X}_i$, the $\mathbf{X}_c$ and $\mathbf{X}_d$ are positions of two different particles chosen randomly from those located in the neighbourhood of $\mathbf{X}_i$ (for a definition of the neighbourhood used we refer the reader to Wang et al. 2013), the $\mathbf{X}_e$ and $\mathbf{X}_f$ are positions of two different particles chosen randomly from the entire swarm and—what is crucial—r_1, r_2, r_3, r_4, r_5 and r_6 are random numbers from the [0,1] interval meeting the following constraints: $r_1 + r_2 + r_3 = 1$ and $r_4 + r_5 + r_6 = 1$. One may note that for given domain $S = [0, 1]^D$, $\forall i \ \forall j \in D \ X_{ij} > 0 \ pbest_{ij} > 0$, but $-1 \leq (X_{cj} - X_{dj}) \leq 1$ and $-1 \leq (X_{ej} - X_{fj}) \leq 1$, the third terms in equations for generating $\mathbf{LX}_i$ and $\mathbf{GX}_i$ may be negative. This leads to a decrease in position values of $\mathbf{LX}_i$ and $\mathbf{GX}_i$ particles and causes the bias. In fact, if no boundary constraints were applied, elements of $\mathbf{LX}_i$ and $\mathbf{GX}_i$ could also be negative. Figure 8 compares the edfs of DNS-PSO and DNS-PSO with neighbourhood search excluded (i.e. when $\mathbf{LX}_i$ and $\mathbf{GX}_i$ are skipped) for selected cases with $PopSize = 100$ and MNFC set to 200, 1000 and 10,000. One can conclude from Fig. 8 that indeed the neighbourhood search seems to be this part of DNS-PSO to which the structural bias can be mostly attributed.

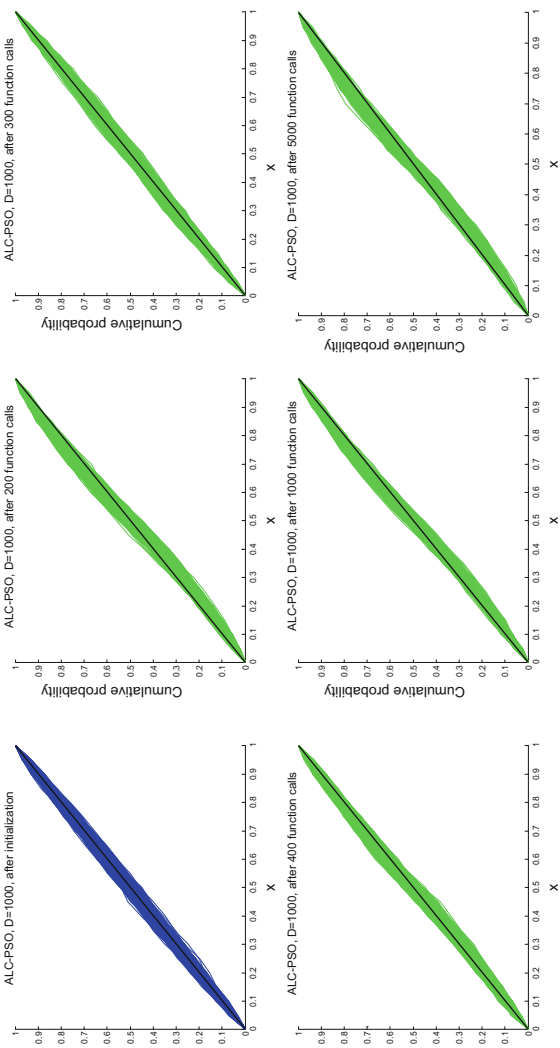

Fig. 9 Empirical distribution function (edf) of the localization of best solutions of 1000-dimensional fully random function found by ALC-PSO algorithm with population size set to 100 individuals after various numbers of function calls. Each *green line* and each *blue line* represent edf for a single axis out of D ($D = 1000$) (Color figure online)

However, we must express our doubt that the justification of the structural bias may not be so easily given in each case. For example, the results presented in Table 8 reveal some strange behaviour of the ALC-PSO algorithm during the first few hundred function calls. The percentage of H0 hypotheses rejected (see Table 8) increased from 5.2 after initialization to 68.1 after 200 function calls, then fell back to 7.9 after 300 function calls, and afterwards slowly increased, similarly to the DE algorithms. To verify that such a result is not accidental, all experiments with 200 and 300 function calls by ALC-PSO were repeated five times with identical parameter settings and each time this similar strange behaviour was observed. These show that some algorithms, under specific circumstances, may exhibit a temporary tendency to concentrate solutions in some parts of the search space. This observation can also be confirmed visually by verifying changes in the edf during the first thousands of function calls (Fig. 9). However, we failed to find any reason for such strange behaviour of ALC-PSO, which may be a warning that the search for the explanation of the structural bias in various algorithms, as well as its practical impact on the results, may not always be as simple as in the case of DNS-PSO.

4.4 Strength of structural bias and algorithms' performance in the case of various benchmark problems

To verify whether the presence of structural bias affects the performance of metaheuristics, all algorithms summarized in Table 1 are applied to solve 22 real-world problems from the CEC2011 set (Das and Suganthan 2010) and 50-dimensional variants of 30 artificial benchmark problems from the CEC2014 set (Liang et al. 2013). The averages of the best objective function values obtained within 150,000 function calls in the case of CEC2011 (value specified in Das and Suganthan 2010) or within 500,000 function calls in the case of CEC2014 [equal to 10,000D, as specified in Liang et al. (2013)] are given in Tables 9 and 10, whereas their respective standard deviations are shown in Supplementary Tables 1 to 2 (available as Supplementary Material in an online version of the paper). The last rows of Tables 9 and 10 present rankings of each algorithm averaged over all 22 CEC2011 problems (Table 9) or 30 CEC2014 problems (Table 10). The ranking was obtained as follows. First, for each problem separately, the algorithms were ranked from the best (rank 1) to the worst (rank 14) according to the 30-run averaged objective function values. Then, ranks achieved by each algorithm were averaged over all 22 (in the case of CEC2011), or all 30 (in the case of CEC2014) problems, giving the values the reader may find in Tables 9 and 10.

One should point out here that the code of GA-MPC, the winner of the CEC2011 competition, has been downloaded from the conference web page (http://www.ntu.edu.sg/home/epnsugan). For tests on CEC2011 and CEC2014 problems, neither the population size nor other values of control parameters were changed in this study, with one exception. For a few CEC2011 problems (codes implemented in MATLAB are also available on http://www.ntu.edu.sg/home/epnsugan), one may obtain "not a number" as the objective function value, even if the solution is within the specified domain. In such a case, the objective function value was set to 0 in the original GA-MPC code. In the present application, this has been modified and, as the CEC2011 problems aim at minimization, a very large number (10^{100}) have been used instead. This may have some impact on the result, as, for instance, in the case of problem F5 from CEC2011, where the initially generated sample often contains positive, negative and "not a number" objective function values. Although the results obtained for GA-MPC based on the mentioned code differ from those published in Elsayed et al. (2011) and their "modified" values available on the CEC2011 web page, this should not be surprising. In the code of GA-MPC, authors of Elsayed et al. (2011) use a seed from the clock and therefore

Table 9 The 30-run averaged fitness function values obtained for CEC2011 real-world problems

Algorithm		PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE-basic	CDE
Structural-bias group		G2	G3	G4	G1	G1	G5	G4
PopSize		20	20	1D, within [10,50]	40	40	5D	100
Problem	*D*							
1	6	1.671E+01	1.066E+01	1.547E+00	1.139E+01	1.133E+01	0.000E+00	0.000E+00
2	30	−7.758E+00	−1.817E+01	−2.097E+01	−2.717E+01	−2.268E+01	−5.884E+00	−1.873E+0
3	1	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05
4	1	1.393E+01	1.394E+01	1.479E+01	**1.378E+01**	1.419E+01	2.023E+01	1.765E+01
5	30	−2.170E+01	−3.267E+01	**−3.516E+01**	−3.359E+01	−3.274E+01	−2.053E+01	−3.277E+0
6	30	−1.666E+01	−2.421E+01	**−2.854E+01**	−2.193E+01	−2.267E+01	−1.503E+01	−2.726E+0
7	20	1.514E+00	1.131E+00	1.079E+00	1.017E+00	9.787E−01	1.706E+00	1.360E+00
8	7	2.200E+02	2.200E+02	2.200E+02	2.220E+02	2.243E+02	2.200E+02	2.200E+02
9	126	3.366E+05	6.297E+05	2.099E+04	3.205E+03	2.827E+05	8.546E+05	2.407E+03
10	12	−1.358E+01	−1.581E+01	−2.042E+01	−1.858E+01	−1.784E+01	−2.043E+01	**−2.169E+01**
11.1	120	1.990E+07	8.421E+05	1.277E+05	9.895E+05	2.741E+05	1.359E+08	**5.189E+04**
11.2	216	2.630E+07	1.908E+07	1.871E+07	1.794E+07	2.413E+07	3.638E+07	1.746E+07
11.3	6	1.546E+04	1.547E+04	1.545E+04	1.547E+04	1.547E+04	1.544E+04	1.545E+04
11.4	13	1.931E+04	1.923E+04	1.907E+04	1.919E+04	1.913E+04	1.940E+04	1.813E+04
11.5	15	4.864E+04	3.296E+04	3.298E+04	3.299E+04	3.310E+04	3.298E+04	3.277E+04
11.6	40	1.374E+05	1.406E+05	1.361E+05	1.391E+05	1.413E+05	1.387E+05	1.340E+05
11.7	140	3.586E+06	2.050E+06	2.088E+06	1.993E+06	1.972E+06	5.592E+07	1.941E+06
11.8	96	1.901E+06	1.127E+06	1.023E+06	9.749E+05	1.006E+06	9.814E+06	9.759E+05
11.9	96	2.574E+06	1.572E+06	1.314E+06	1.416E+06	1.238E+06	1.067E+07	1.432E+06
11.10	96	1.929E+06	1.178E+06	1.022E+06	9.690E+05	1.001E+06	9.711E+06	9.761E+05
12	26	2.741E+01	1.778E+01	1.547E+01	**1.254E+01**	1.662E+01	2.787E+01	1.971E+01
13	22	2.958E+01	2.379E+01	1.789E+01	2.208E+01	2.346E+01	2.468E+01	1.601E+01
Average ranks		11.48	9.25	6.30	7.32	8.59	11.18	4.95

Table 9 continued

Algorithm		DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Structural-bias group		G3	G3	G3	G2	G2	G5	G6
PopSize		10D	100	100	1D, within [10,50]	90	D+1	1
Problem	*D*							
1	6	8.222E+00	3.318E+00	3.322E−01	1.078E+01	5.939E−01	2.031E+01	2.002E+01
2	30	−1.187E+01	−2.489E+01	−2.108E+01	−2.523E+01	**−2.723E+01**	−2.673E+01	−1.523E+01
3	1	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05	1.151E−05
4	1	2.047E+01	1.596E+01	1.694E+01	1.557E+01	1.412E+01	1.413E+01	1.382E+01
5	30	−2.199E+01	−3.311E+01	−3.402E+01	−3.455E+01	−3.425E+01	−3.449E+01	−3.397E+01
6	30	−1.924E+01	−2.429E+01	−2.845E+01	−2.583E+01	−2.739E+01	−2.330E+01	−2.433E+01
7	20	1.481E+00	1.132E+00	1.232E+00	9.680E−01	9.992E−01	**7.794E−01**	1.247E+00
8	7	2.200E+02	2.200E+02	2.200E+02	2.226E+02	2.200E+02	2.200E+02	2.365E+02
9	126	8.614E+03	2.088E+03	2.214E+03	4.809E+03	2.431E+03	2.253E+06	**1.791E+03**
10	12	−2.008E+01	−2.156E+01	−2.159E+01	−1.971E+01	−2.166E+01	−1.974E+01	−1.020E+01
11.1	120	7.540E+04	5.251E+04	5.237E+04	5.709E+04	5.229E+04	2.503E+05	8.893E+04
11.2	216	1.837E+07	1.743E+07	1.747E+07	1.772E+07	**1.079E+06**	4.565E+07	1.773E+07
11.3	6	1.546E+04	1.545E+04	1.544E+04	1.548E+04	1.544E+04	1.548E+04	1.546E+04
11.4	13	1.814E+04	**1.810E+04**	1.811E+04	1.875E+04	1.832E+04	1.841E+04	1.839E+04
11.5	15	**3.276E+04**	3.278E+04	3.279E+04	3.304E+04	3.278E+04	3.300E+04	3.309E+04
11.6	40	1.333E+05	1.292E+05	1.297E+05	1.355E+05	1.371E+05	**1.280E+05**	1.428E+05
11.7	140	2.066E+06	1.916E+06	**1.906E+06**	2.000E+06	2.150E+06	1.396E+07	2.275E+06
11.8	96	1.259E+06	**9.385E+05**	9.402E+05	9.548E+05	1.048E+06	2.460E+06	1.812E+06
11.9	96	1.756E+06	**9.506E+05**	1.037E+06	1.090E+06	1.250E+06	3.090E+06	2.555E+06
11.10	96	1.252E+06	**9.388E+05**	9.404E+05	9.563E+05	1.045E+06	2.378E+06	1.625E+06
12	26	1.622E+01	1.595E+01	1.750E+01	1.486E+01	1.324E+01	2.154E+01	4.975E+01
13	22	1.742E+01	1.867E+01	1.669E+01	2.118E+01	**1.070E+01**	2.647E+01	3.613E+01
Average ranks		7.98	4.16	4.11	6.18	4.52	9.39	9.59

The best results are bolded

Table 10 The 30-run averaged fitness function values obtained for 50-dimensional variants of CEC2014 benchmark problems

Algorithm	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE- basic	CDE
Structural-bias group	G2	G3	G4	G1	G1	G5	G4
PopSize	20	20	1D, within[10, 50]	40	40	5D	100
Problem							
1	2.304E+08	2.140E+06	1.526E+07	8.947E+05	6.993E+06	5.924E+08	2.647E+06
2	1.573E+10	1.156E+05	3.872E+01	7.042E+03	1.113E+06	5.280E+09	1.039E+03
3	3.687E+04	1.622E+04	2.120E+03	1.617E+03	5.897E+02	1.344E+05	1.394E+02
4	1.057E+03	9.872E+01	9.823E+01	8.125E+01	1.953E+02	7.136E+02	9.427E+01
5	2.113E+01	2.099E+01	2.046E+01	2.112E+01	2.096E+01	2.112E+01	2.060E+01
6	5.185E+01	3.605E+01	2.795E+01	3.329E+01	2.838E+01	6.532E+01	1.790E+01
7	1.479E+02	6.924E−02	3.199E−04	9.591E−03	8.007E−03	2.944E+01	1.112E−03
8	3.861E+02	3.902E−02	1.315E−13	1.898E+02	8.531E+01	3.289E+02	0.000E+00
9	4.342E+02	2.195E+02	1.187E+02	2.282E+02	1.430E+02	4.669E+02	1.319E+02
10	1.292E+04	3.316E+02	**2.555E+00**	4.107E+03	2.374E+03	9.905E+03	4.122E+01
11	1.306E+04	5.389E+03	4.582E+03	6.219E+03	5.611E+03	1.312E+04	7.677E+03
12	3.392E+00	9.989E−01	3.407E−01	2.502E+00	7.517E−01	3.174E+00	8.004E−01
13	1.220E+00	5.971E−01	3.996E−01	5.776E−01	5.491E−01	8.004E−01	3.875E−01
14	4.141E+01	5.765E−01	3.029E−01	3.858E−01	6.459E−01	3.993E+00	**2.963E−01**
15	3.053E+03	3.566E+01	1.799E+01	2.125E+01	1.572E+01	3.973E+03	1.416E+01

Table 10 continued

Algorithm	PSO-basic	ALC-PSO	CLPSO	DNS-PSO	PSO-init-weight	DE- basic	CDE
Structural-bias group	G2	G3	G4	G1	G1	G5	G4
PopSize	20	20	1D, within[10, 50]	40	40	5D	100
16	2.195E+01	2.105E+01	1.860E+01	2.117E+01	2.012E+01	2.244E+01	1.964E+01
17	1.141E+07	4.268E+05	2.415E+06	1.570E+05	1.409E+06	2.678E+07	3.231E+05
18	4.759E+08	2.460E+03	**1.440E+02**	1.001E+03	1.226E+04	1.091E+06	3.772E+02
19	1.392E+02	2.918E+01	1.773E+01	1.633E+01	5.750E+01	4.765E+01	3.473E+01
20	6.747E+03	5.192E+03	6.251E+03	4.519E+02	5.335E+02	4.868E+04	7.014E+02
21	4.788E+06	2.330E+05	1.506E+06	1.032E+05	3.545E+05	1.117E+07	2.958E+05
22	1.592E+03	1.257E+03	5.782E+02	1.108E+03	8.023E+02	1.425E+03	5.941E+02
23	4.889E+02	3.440E+02	3.440E+02	**2.000E+02**	3.462E+02	3.482E+02	3.440E+02
24	3.663E+02	2.769E+02	2.580E+02	**2.000E+02**	2.743E+02	3.324E+02	2.680E+02
25	2.515E+02	2.157E+02	2.153E+02	2.000E+02	2.206E+02	3.059E+02	2.007E+02
26	2.013E+02	1.482E+02	1.005E+02	1.961E+02	1.716E+02	1.008E+02	1.004E+02
27	1.646E+03	1.395E+03	7.626E+02	**2.428E+02**	1.126E+03	1.822E+03	4.856E+02
28	3.634E+03	3.273E+03	1.388E+03	**2.126E+02**	2.786E+03	1.748E+03	1.275E+03
29	4.474E+07	6.774E+06	1.396E+03	2.936E+06	5.216E+07	8.651E+05	1.825E+03
30	2.257E+05	2.270E+04	9.905E+03	1.241E+04	4.697E+04	1.475E+05	8.986E+03
Average ranks	13.32	9.41	5.68	8.59	8.86	13.05	6.20

Table 10 continued

Algorithm	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Structural-bias group	G3	G3	G3	G2	G2	G5	G6
PopSize	10D	100	100	1D, within [10, 50]	90	D+1	1
Problem							
1	4.056E+05	7.704E+04	1.461E+05	1.387E+05	3.963E+05	**5.181E+04**	2.836E+05
2	**0.000E+00**	3.734E+03	6.969E−10	3.652E−05	3.727E+03	9.540E+03	8.092E+03
3	**0.000E+00**	7.845E−03	3.000E+02	2.296E−03	1.404E+01	5.630E−07	8.917E−06
4	2.600E+01	3.453E+01	**1.951E+01**	5.308E+01	6.359E+01	3.235E+01	4.670E+01
5	2.113E+01	2.050E+01	2.060E+01	2.075E+01	2.024E+01	2.000E+01	2.000E+01
6	**3.439E+00**	5.805E+00	1.014E+01	3.181E+01	5.055E+00	2.759E+01	8.137E+01
7	1.208E−03	6.546E−03	4.301E−03	3.326E−02	3.427E−03	1.358E−02	**7.023E−11**
8	2.747E+02	5.557E+01	0.000E+00	8.455E+01	5.222E+01	1.629E+02	5.299E+02
9	3.367E+02	1.277E+02	8.703E+01	1.427E+02	**5.651E+01**	1.532E+02	7.983E+02
10	1.199E+04	1.533E+03	1.349E+01	1.739E+02	1.141E+03	6.737E+03	6.773E+03
11	1.287E+04	8.089E+03	6.474E+03	6.021E+03	**3.657E+03**	6.916E+03	7.011E+03
12	3.324E+00	7.290E−01	7.427E−01	7.693E−01	**8.171E−02**	2.174E−01	1.057E+00
13	3.464E−01	**3.046E−01**	3.774E−01	5.291E−01	4.241E−01	7.607E−01	3.283E−01
14	3.399E−01	3.221E−01	3.034E−01	3.487E−01	3.346E−01	3.061E−01	3.018E−01
15	2.849E+01	9.405E+00	1.397E+01	3.602E+01	**5.621E+00**	3.000E+01	7.344E+02

Table 10 continued

Algorithm	DEGL	MDE-pBX	SADE	CLPSO-DEGL	GA-MPC	NMA	RA
Structural-bias group	G3	G3	G3	G2	G2	G5	G6
PopSize	10D	100	100	1D, within [10, 50]	90	D+1	1
16	2.153E+01	1.979E+01	1.921E+01	1.908E+01	**1.704E+01**	2.195E+01	2.371E+01
17	**1.753E+03**	4.466E+03	1.071E+04	1.085E+04	2.973E+04	2.833E+03	1.600E+04
18	1.802E+02	3.245E+02	3.222E+02	1.010E+03	4.799E+02	2.343E+02	2.602E+03
19	1.272E+01	1.056E+01	1.669E+01	1.779E+01	**5.767E+00**	3.249E+01	2.840E+01
20	**9.189E+01**	1.698E+02	2.864E+02	2.058E+02	2.119E+02	5.474E+02	7.217E+02
21	**6.417E+02**	1.401E+03	4.551E+03	2.756E+03	1.684E+04	3.110E+03	1.707E+04
22	1.139E+03	7.680E+02	**4.013E+02**	4.925E+02	1.010E+03	1.145E+03	1.242E+03
23	3.440E+02	3.440E+02	3.440E+02	3.440E+02	3.370E+02	3.440E+02	3.444E+02
24	2.750E+02	2.781E+02	2.739E+02	2.821E+02	2.685E+02	2.830E+02	3.016E+02
25	2.060E+02	2.102E+02	2.096E+02	2.311E+02	2.005E+02	2.000E+02	2.208E+02
26	1.004E+02	1.043E+02	1.316E+02	1.143E+02	1.004E+02	1.415E+02	1.219E+02
27	4.696E+02	5.568E+02	5.838E+02	1.216E+03	1.239E+03	9.970E+02	1.565E+03
28	1.149E+03	1.165E+03	1.176E+03	2.416E+03	3.695E+02	6.928E+03	7.805E+03
29	8.312E+05	9.349E+02	1.008E+03	1.154E+03	**2.158E+02**	1.283E+03	1.205E+04
30	5.618E+03	9.436E+03	1.048E+04	1.338E+04	**1.143E+03**	2.155E+04	1.430E+04
Average ranks	6.18	4.50	4.02	6.05	4.41	6.55	8.18

The best results are bolded

different results are expected when the computations are repeated—this is also discussed on the CEC2011 homepage (http://www.ntu.edu.sg/home/epnsugan).

The rankings of algorithms on the CEC2011 problems presented in Table 9 indicate that the performance of CDE and CLPSO, the two modern metaheuristics for which the presence of structural bias turned out to be inconclusive, is moderate. Their averaged rankings are poorer than those of three structurally biased methods, namely SADE, GA-MPC and MDE_pBX, but the differences in rankings among these five best algorithms are small and (see Supplementary Table 3) statistically insignificant at the $\alpha = 0.05$ level. On the other hand, two structurally unbiased methods, the basic version of the DE algorithm (DE-basic) and the historical NMA algorithm turned out to be among the four poorest optimizers. The two most biased algorithms (PSO-init-weight and DNS-PSO) are in the middle of the pack. According to the statistical tests performed for the results obtained for the CEC2011 problems (Supplementary Table 3, see also the discussion at the end of Sect. 3), the differences between most of the tested algorithms are not significant at the $\alpha = 0.05$ level and one cannot find any direct link between the presence of an algorithm's structural bias and an algorithm's performance. Interestingly, the differences between structurally unbiased NMA and DE-basic on the one hand, and the four best methods (SADE, GA-MPC, MDE_pBX and CDE, among which the three best ones are structurally biased) on the other, are statistically significant.

The ranking of the algorithms on the 50-dimensional artificial benchmark problems from CEC2014 looks similar to the one obtained for CEC2011 real-world problems (see Table 10). Algorithms SADE, GA-MPC and MDE_pBX (all structurally biased) remain the best three approaches, while structurally (almost) unbiased CLPSO and CDE came 4th and 7th, respectively. There is, however, one major difference: structurally unbiased historical NMA turned out to be quite highly ranked, and the difference between NMA and best performing SADE is not statistically significant at the $\alpha = 0.05$ level (see Supplementary Table 4). DE-basic remains among the two poorest methods. Although, again, no clear link between the strength of structural bias and the performance of the algorithm is observed, nonetheless the three out of four supposedly unbiased methods perform quite well.

According to the above results, the practical implications of structural bias of metaheuristics remain questionable. It looks as though the effect from the fitness function landscape was much higher than the effect of structural bias. However, the presence of structural bias should not be ignored. Firstly, we cannot exclude the possibility that the structural bias indeed hampers the search, preventing algorithms from achieving better results. Would the variants of SADE, GA-MPC and MDE_pBX perform better if the part responsible for the presence of structural bias was detected and removed? Secondly, algorithms that show the strongest indications of structural bias perform rather poorly on the CEC2011 and CEC2014 problems. Thirdly, three out of four methods that are almost structurally unbiased (CLPSO, CDE and NMA) show quite good performance: CLPSO is among the best PSO variants; CDE is ranked in the middle of the DE-based ones; NMA remains competitive against 50 years younger optimizers on artificial benchmark problems from CEC2014 and at least for some real-world problems from CEC2011 (indeed, with some surprise, we must acknowledge that NMA turned out to be the best approach for two CEC2011 problems, see Table 9, although it performed poorly on the majority of the others). Hence, one can conclude that although the presence of "moderate" structural bias does not have to prevent potentially good performance of the optimizer, methods that are unbiased may perform better than expected (like NMA, or also relatively old CLPSO).

In this paper, we did not intend to discuss any performance comparison between PSO and DE approaches. The number of variants from both families of methods is too large, and many classical or the most recent ones were not considered in this study. We believe that no strong

opinions regarding the overall performance of PSO or DE algorithms can be formulated based on this research. The sole purpose of this section was to verify whether there is any relationship between the presence of structural bias and algorithm performance.

5 Conclusions

This paper's goal was to empirically verify the presence of structural bias in 11 particle swarm optimization (PSO) and differential evolution (DE) algorithms. For comparison, tests were also performed on three other optimizers, namely a genetic algorithm, GA-MPC (Elsayed et al. 2011) which was the winner of the CEC2011 competition, and two historical direct search methods, namely the Nelder and Mead (1965) and Rosenbrock's (1960) algorithms.

In the vast majority of tested PSO and DE variants, the presence of structural bias was detected. In addition, GA-MPC was found to be structurally biased. However, some PSO (CLPSO) (Liang et al. 2006) and DE (CDE) (Cai et al. 2011) algorithms turned out to be almost structurally unbiased, at least when they were applied with the appropriate population size. The historical Nelder and Mead (1965) algorithm and the basic DE variant proposed by Storn and Price (1995) are the only methods among the 14 algorithms tested that showed no sign of structural bias. We also found that testing some optimization methods (i.e. Rosenbrock's 1960 algorithm) on a fully random function, as proposed in Kononova et al. (2015), may not allow us to draw any conclusions regarding the presence of structural bias. In general, the PSO algorithms seem to be more structurally biased than the DE algorithms, probably because the initial version of PSO (Eberhart and Kennedy 1995) is structurally biased, contrary to the initial version of DE (Storn and Price 1995).

The impact of the population size, problem dimensionality and the number of allowed function calls on the strength of structural bias depends on the particular algorithm and no general conclusion may be drawn. The ALC-PSO method (Chen et al. 2013) can serve as a good illustration of this fact. For example, the strength of structural bias depends on the population size and on the length of the run, but not monotonically: when only 10,000 function calls are allowed, the larger the population size, the stronger the structural bias of ALC-PSO; but when the number of function calls is extended to 300,000, the opposite relation is found. Various algorithm-specific control parameters may affect the presence and the strength of structural bias as well. In the paper, we pointed out how to (almost) remove the structural bias from the most biased algorithm [DNS-PSO (Wang et al. 2013)].

The presence of structural bias affects the distribution of the best solutions very quickly, often in the first generations in the case of PSO, and during the first dozens of generations in the case of DE algorithms. What is quite strange and not understood at this point is the following finding: in the case of some DE algorithms, the effect of structural bias may start receding after a large enough number of function calls, although it does not completely disappear.

To find the link between the presence of structural bias and algorithms' performance, all 14 methods were tested on 22 real-world problems proposed for CEC2011, and 50-dimensional versions of 30 artificial benchmarks proposed for CEC2014 competitions. No firm relationship was found. The best methods were those with a moderate structural bias. The performance of two PSO and DE variants that are (almost) structurally unbiased (CLPSO and CDE) turned out to be slightly above the middle of the pack. The structurally unbiased basic DE variant performed poorly. However, the unbiased 50-year-old Nelder–Mead algorithm, although it turned out rather poorly on most CEC2011 real-world problems, enjoyed a better

performance than all 13 other optimizers on two among the CEC2011 problems, and was highly ranked on the CEC2014 artificial benchmark problems. One may speculate that the impact of the structural bias on the search is relatively low compared to the impact of the fitness landscape, although this may largely depend on the specific problem to be solved. To find out to what extent structural bias affects the search on various types of fitness landscapes may be an important topic of future studies. Nonetheless, we highly recommend verifying the presence of structural bias in the currently existing and newly proposed algorithms and removing it, if possible.

Acknowledgements This work was supported within statutory activities No 3841/E-41/S/2015 of the Ministry of Science and Higher Education of Poland. The authors would like to thank Prof. Ponnuthurai N. Suganthan for providing the MATLAB codes of MDE_pBX and CLPSO algorithms, as well as the authors of Elsayed et al. (2011) and Ulas et al. (2012) papers for making the code of the GA-MPC algorithm and post-hoc statistical procedures widely available on the web. The authors are very grateful for the very detailed comments from the reviewers and editors that turned out very useful during preparation of the final version of the paper.

Compliance with ethical standards

Conflict of interest Adam Piotrowski has benefited from a grant of the Ministry of Science and Higher Education (MNiSW) of Poland. Jaroslaw Napiorkowski declares no conflict of interest.

References

Ampellio, E., & Vassio, L. (2016). A hybrid swarm-based algorithm for single-objective optimization problems involving high-cost analyses. *Swarm Intelligence*, *10*, 99–121.

Auger, A., & Doerr, B. (2011). *Theory of randomized search heuristics. Vol. 1 of theoretical computer science*. Singapore: World Scientific.

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, *7*, 109–124.

Bonyadi, M. R., & Michalewicz, Z. (2014). A locally convergent rotationally invariant particle swarm optimization algorithm. *Swarm Intelligence*, *8*, 159–198.

Bonyadi, M. R., & Michalewicz, Z. (2016). Particle swarm optimization for single objective continuous space problems: A review. *Evolutionary Computation*. doi:10.1162/EVCO_r_00180.

Boussad, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117.

Cai, Z. H., Gong, W. Y., Ling, C. X., & Zhang, H. (2011). A clustering-based differential evolution for global optimization. *Applied Soft Computing*, *11*(1), 1363–1379.

Chen, W. N., Zhang, J., Lin, Y., Chen, N., Zhan, Z. H., Chung, H. S. H., et al. (2013). Particle swarm optimization with an aging leader and challengers. *IEEE Transactions on Evolutionary Computation*, *17*(2), 241–258.

Chinta, S., Kommadath, R., & Kotecha, P. (2016). A note on multi-objective improved teaching-learning based optimization algorithm (MO-ITLBO). *Information Sciences*, *373*, 337–350.

Cleghorn, C. W., & Engelbrecht, A. P. (2014). A generalized theoretical deterministic particle swarm model. *Swarm Intelligence*, *8*, 35–59.

Cleghorn, C. W., & Engelbrecht, A. P. (2015). Particle swarm variants: Standardized convergence analysis. *Swarm Intelligence*, *9*, 177–203.

Clerc, M. (2006). *Particle swarm optimization*. London: ISTE Ltd.

Clerc, M., & Kennedy, J. (2002). The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73.

Črepinšek, M., Liu, S. H., & Mernik, L. (2012). A note on teaching-learning-based optimization algorithm. *Information Sciences*, *212*, 79–93.

Črepinšek, M., Liu, S. H., & Mernik, M. (2014). Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them. *Applied Soft Computing*, *19*, 161–170.

Črepinšek, M., Liu, S. H., Mernik, L., & Mernik, M. (2016). Is a comparison of results meaningful from the inexact replications of computational experiments? *Soft Computing*, *20*(1), 223–235.

Das, S., Abraham, A., & Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives. In *Studies in computational intelligence* (Vol. 116). Berlin: Springer.

Das, S., Abraham, A., Chakraboty, U. K., & Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, *13*(3), 526–553.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution—An updated survey. *Swarm and Evolutionary Computation*, *27*, 1–30.

Das, S., & Suganthan, P. N. (2010). *Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems*. Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India, Technical Report, Dec. 2010.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 27–54.

Derrac, J., Garcia, S., Hui, S., Suganthan, P. N., & Herrera, F. (2014). Analyzing convergence performance of evolutionary algorithms: A statistical approach. *Information Sciences*, *289*, 41–58.

Draa, A. (2015). On the performances of the flower pollination algorithm—Qualitative and quantitative analyses. *Applied Soft Computing*, *34*, 349–371.

Droste, S., Jansen, T., & Wegener, I. (2002). Optimization with randomized search heuristics—The (A)NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, *287*(1), 131–144.

Droste, S., Jansen, T., & Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, *39*, 525–544.

Duéñez-Guzmán, E. A., & Vose, M. D. (2013). No free lunch and benchmarks. *Evolutionary Computation*, *21*(2), 293–312.

Dymond, A. S., Engelbrecht, A. P., Kok, S., & Heyns, P. S. (2015). Tuning optimization algorithms under multiple objective function evaluation budgets. *IEEE Transactions on Evolutionary Computation*, *19*(3), 341–358.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th international symposium on micro machine and human science* (pp. 39–43). Institute of Electrical and Electronics Engineers (IEEE).

Eiben, A. E., & Jelasity, M. (2002). A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress* (CEC2002) (Vol. 1, pp. 582–587). Institute of Electrical and Electronics Engineers (IEEE).

Eiben, A. E., & Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, *521*, 476–482.

Elsayed, S. M., Sarker, R. A. & Essam, D. L. (2011). GA with a new multi-parent crossover for constrained optimization. In *IEEE Congress on evolutionary computation 2011* (pp. 857–864). Institute of Electrical and Electronics Engineers (IEEE).

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2014). A new genetic algorithm for solving optimization problems. *Engineering Applications of Artificial Intelligence*, *27*, 57–69.

Epitropakis, M. G., Plagianakos, V. P., & Vrahatis, M. N. (2012). Evolving cognitive and social experience in particle swarm optimization through differential evolution: A hybrid approach. *Information Sciences*, *216*, 50–92.

Fister, I, Jr., Maklar, U., Brest, J., & Fister, I. (2016). A new population-based nature-inspired algorithm every month: Is the current era coming to the end? In *Proceedings of the 3rd Student Computer Science Research Conference* (pp. 33–37). Ljubljana: Slovenia, University of Primorska Press.

Fogel, D. B. (2000). What is evolutionary computation? *IEEE Spectrum*, *37*(2), 26–32.

Fong, S., Wang, X., Xu, Q., Wong, R., Fiaidhi, J., & Mohammed, S. (2016). Recent advances in metaheuristic algorithms: Does the Makara dragon exist? *The Journal of Supercomputing*, *72*, 3764–3786.

Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study of differential evolution. In *Advances in intelligent systems, fuzzy systems, evolutionary computation*. Interlaken: WSEAS Press.

Garcia, S., & Herrera, F. (2008). An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Hall, J. C., Mills, B., Nguyen, H., & Hall, J. L. (1996). Methodologic standards in surgical trials. *Surgery*, *119*(4), 466–472.

Helwig, S., Branke, J., & Mostaghim, S. (2013). Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *17*(2), 259–271.
Holland, I. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.
Hu, Z. B., Su, Q. H., Yang, X. S., & Xiong, Z. G. (2016). Not guaranteeing convergence of differential evolution on a class of multimodal functions. *Applied Soft Computing*, *41*, 479–487.
Hu, Z. B., Xiong, S. W., Su, Q. H., & Fang, Z. X. (2014). Finite Markov chain analysis of classical differential evolution algorithm. *Journal of Computational and Applied Mathematics*, *268*, 121–134.
Igel, C., & Toussaint, M. (2003). On classes of functions for which no free lunch results hold. *Information Processing Letters*, *86*(6), 317–321.
Igel, C., & Toussaint, M. (2004). A no-free lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, *3*, 313–322.
Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2012). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part B - Cybernetics*, *42*(2), 482–500.
Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, *8*, 687–697.
Kennedy, J., & Eberhart, R. C. (1995). Particle swam optimization. In *Proceedings of the IEEE international conference on neural networks* (pp. 1942–1948). Institute of Electrical and Electronics Engineers (IEEE), 1995.
Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., et al. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, *28*(8), 721–734.
Kolda, T. G., Lewis, R. A., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, *45*(3), 385–482.
Kononova, A. V., Corne, D. W., De Wilde, P., Shneer, V., & Caraffini, F. (2015). Structural bias in population-based algorithms. *Information Sciences*, *298*, 468–490.
Köppen, M., Wolpert, D. H., & Macready, W. G. (2001). Remarks on a recent paper on the " No free lunch" theorems. *IEEE Transactions on Evolutionary Computation*, *5*(3), 295–296.
Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal of Optimization*, *9*(1), 112–147.
Leonard, B. J., Engelbrecht, A. P., & Cleghorn, C. W. (2015). Critical considerations on angle modulated particle swarm optimizers. *Swarm Intelligence*, *9*, 291–314.
Liang, J. J., Qu, B. Y., & Suganthan, P. N. (2013). *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization*. Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore.
Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295.
Liao, T., Aydin, D., & Stützle, T. (2013). Artificial bee colonies for continuous optimization: Experimental analysis and improvements. *Swarm Intelligence*, *7*, 327–356.
Malan, K. M., & Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, *241*, 148–163.
Malan, K. M., & Engelbrecht, A. P. (2014). Characterising the searchability of continuous optimisation problems for PSO. *Swarm Intelligence*, *8*, 275–302.
Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modelling and Computer Simulation*, *8*, 3–30.
Mernik, M., Liu, S. H., Karaboga, D., & Črepinšek, M. (2015). On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation. *Information Sciences*, *291*, 115–127.
Michalewicz, Z. (2012). Quo Vadis, evolutionary computation? On a growing gap between theory and practice. In *Proceedings of the 2012 World Congress conference on advances in computational intelligence (WCCI'12)* (pp. 98–121). Berlin: Springer.
Muñoz, A. A., & Smith-Miles, K. A. (2016). Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary Computation*. doi:10.1162/EVCO_a_00194.
Muñoz, M. A., Sun, Y., Kirley, M., & Halgamuge, S. K. (2015). Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, *317*, 224–245.
Nelder, J. A., & Mead, R. (1965). A simplex-method for function minimization. *Computer Journal*, *7*(4), 308–313.

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, *33*(1–2), 61–106.
Piotrowski, A. P. (2015). Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. *Information Sciences*, *297*, 191–201.
Piotrowski, A. P. (2016). Review of differential evolution population size. *Swarm and Evolutionary Computation*. doi:10.1016/j.swevo.2016.05.003.
Piotrowski, A. P., & Napiorkowski, M. J. (2016). May the same numerical optimizer be used when searching either for the best or for the worst solution to a real-world problem? *Information Sciences*, *373*, 124–148.
Piotrowski, A. P., Napiorkowski, J. J., & Rowinski, P. M. (2014). How novel is the "novel" black hole optimization approach? *Information Sciences*, *267*, 191–200.
Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. An overview. *Swarm Intelligence*, *1*, 33–57.
Pošik, P., Huyer, W., & Pal, L. (2012). A comparison of global search algorithms for continuous black box optimization. *Evolutionary Computation*, *20*(4), 509–541.
Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (2006). *Numerical recipes in FORTRAN 77: The art of scientific computing*. Cambridge: Cambridge University Press.
Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution. A practical approach to global optimization*. Berlin: Springer.
Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*(2), 398–417.
Rada-Vilela, J., Johnston, M., & Zhang, M. (2014). Deception, blindness and disorientation in particle swarm optimization applied to noisy problems. *Swarm Intelligence*, *8*, 247–273.
Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog.
Rosenbrock, H. H. (1960). An automated method for finding the greatest or least value of a function. *Computer Journal*, *3*(3), 175–184.
Rowe, J. E., Vose, M. D., & Wright, A. H. (2009). Reinterpreting no free lunch. *Evolutionary Computation*, *17*(1), 117–129.
Salcedo-Sanz, S. (2016). Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Physics Reports*, *65*, 1–70.
Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation on benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, *39*, 263–278.
Schumacher, C., Vose, M. D., & Whitley, L. D. (2001). The no free lunch and problem description length. In *Proceedings of the 2001 genetic and evolutionary computation conference* (pp. 565–570). Morgan Kaufmann.
Shaffer, J. P. (1986). Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, *81*(395), 826–831.
Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceeding in IEEE Congress on Evolutionary Computation (CEC1998)* (pp. 69–73). Institute of Electrical and Electronics Engineers (IEEE).
Simon, D., Rarick, R., Ergezer, M., & Du, D. W. (2011). Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. *Information Sciences*, *181*, 1224–1248.
Sörensen, K. (2015). Metaheuristics—The metaphor exposed. *International Transactions in Operational Research*, *22*, 3–18.
Sörensen, K., Sevaux, M., & Glover, F. (2015). A history of Metaheuristics. In*Proceedings of ORBEL29—29th Belgian conference on Operations Research*.
Stephens, M. A. (1970). Use of the Kolmogorov–Smirnov, Cramer–von Mises and related statistics without extensive tables. *Journal of the Royal Statistical Society Series B - Statistical Methodology*, *32*, 115–122.
Storn, R., & Price, K. V. (1995). *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.
Storn, R., & Price, K. V. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., et al. (2005). *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.
Ulas, A., Yildiz, O. T., & Alpaydin, E. (2012). Cost-conscious comparison of supervised learning algorithms over multiple data sets. *Pattern Recognition*, *45*, 1772–1781.

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 225–239.

Veček, N., Mernik, M., & Črepinšek, M. (2014). A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms. *Information Sciences*, *277*, 656–679.

Wang, H., Sun, H., Li, C., Rahnamayan, S., & Pan, J. S. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences*, *223*, 119–135.

Weise, T., Chiong, R., & Tang, K. (2012). Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, *27*(5), 907–936.

Weyland, D. (2010). A rigorous analysis of the harmony search algorithm—How the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing*, *1–2*, 50–60.

Weyland, D. (2015). A critical analysis of the harmony search algorithm—How not to solve sudoku. *Operations Research Perspectives*, *2*, 97–105.

Whitley, D., & Rowe, J. (2008). Focused no free lunch theorems. In *Proceedings of the 10th annual conference on genetic and evolutionary computation* (pp. 811–818). Association for Computing Machinery (ACM), 2008.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Xin, B., Chen, J., Zhang, J., Fang, H., & Peng, Z. H. (2012). Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy. *IEEE Transactions on Systems, Mac and Cybernetics, Part C - Applications and Reviews*, *42*(5), 744–767.

Xing, B., & Gao, W. J. (2014). *Innovative computational intelligence: A rough guide to 134 clever algorithms. Intelligent Systems Reference Library* (Vol. 62). Berlin: Springer.

Yancey, J. M. (1990). Ten rules for reading clinical research reports. *The American Journal of Surgery*, *159*(6), 533–539.

Yao, X., Liu, Y., & Li, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, *3*(2), 82–102.

Yuen, S. Y., & Zhang, X. (2015). On composing an algorithm portfolio. *Memetic Computing*, *7*, 203–214.

Zelinka, I. (2015). A survey on evolutionary algorithms dynamics and its complexity—Mutual relations, past, present and future. *Swarm and Evolutionary Computation*, *25*, 2–14.

Zhang, Y. D., Wang, S. H., & Ji, G. L. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*. Article ID: 931256.