Soft Computing manuscript No.
(will be inserted by the editor)

On the use of single non-uniform mutation in lightweight metaheuristics

Souheila Khalfi · Giovanni Iacca[*] · Amer Draa

Received: date / Accepted: date

Abstract This paper introduces two novel lightweight algorithms based on a Single Non-Uniform Mutation (SNUM) operator: a single solution algorithm, and a SNUM-based compact Genetic Algorithm. The first algorithm, called also SNUM with reference to the operator, performs the search by an iterative process that perturbs one design variable selected randomly from a single solution. The latter, called compact SNUM (cSNUM), incorporates the SNUM mechanism into the compact Genetic Algorithm scheme, that replaces a population of solutions with a probabilistic model. Both approaches are characterised by a purposely simple and highly generic algorithmic structure. These two attractive features make it possible to readily employ the core part of each algorithm and combine it with other techniques for extended complexity. The results obtained from applying the two proposed algorithms on the BBOB and CEC-2017 benchmarks reveal that the use of SNUM is largely beneficial. Not only the two algorithms (in particular cSNUM) are able to deal with separable functions, especially when the problem dimensionality increases, but they also prove to be competitive on other classes of functions, displaying very good performances compared to other methods from the literature, also on non-separable functions.

[*] Corresponding author

Souheila Khalfi
Department of Fundamental Informatics and its Applications, Constantine 2 University
Nouvelle Ville Ali Mendjeli, 25016, Algeria
E-mail: souheila.khalfi@univ-constantine2.dz
ORCID: 0000-0002-5033-8937

Giovanni Iacca
Department of Information Engineering and Computer Science, University of Trento
Via Sommarive 9, 38123 Povo (TN), Italy
Tel: +39 0461 28 5220
E-mail: giovanni.iacca@unitn.it
ORCID: 0000-0001-9723-1830

Amer Draa
Department of Fundamental Informatics and its Applications, Constantine 2 University
Nouvelle Ville Ali Mendjeli, 25016, Algeria
E-mail: amer.draa@univ-constantine2.dz
ORCID: 0000-0003-2448-1286

Keywords single solution metaheuristic · compact optimisation · lightweight metaheuristic · limited-memory algorithm · non-uniform mutation.

1 Introduction

As the context of computer science and engineering continuously changes, there is a strong need for efficient numerical optimisation approaches. These algorithms are becoming ever more complex, in order to face a large number of hard-to-solve real-world optimisation problems [68]. Challenges in modern optimisation problems may result, for instance, from the large-scale nature of some applications [45], the specific properties of the objective function [23,24], real-time demands [48,54], or limited hardware resources [28,29]. In these cases, solving optimisation problems with exact methods within reasonable time can prove to be either ineffective or unachievable. Hence, the use of metaheuristics has emerged as a viable alternative [56].

The field of metaheuristics, which are mainly based on Swarm and Evolutionary Computation [4], is constantly expanding. In this sense, *population-based algorithms* (i.e., algorithms that keep a pool of candidate solutions, which are manipulated at every step to generate new ones) represent the basis of most of the existing literature. Typical examples of this kind of algorithms are Genetic Algorithms [4] and Particle Swarm Optimisation [39]. It is generally admitted that population-based techniques can achieve better results compared to *single solution algorithms* (i.e., algorithms that manipulate just one single candidate solutions at a time, e.g. Simulated Annealing [40]). This is mainly because population-based algorithms conduct an implicitly parallel search and maintain a higher solution diversity [55]. However, population-based algorithms are often characterised by complex search logics which require more memory resources and computational power than single solution counterparts [37]. In addition, complex algorithms are usually hard to control and analyse (since the effect of their search operators may be hard to characterise), and, most importantly, not always they provide better performances, as shown e.g. in [37,9].

Compared to population-based algorithms, single solution metaheuristics have instead (in general) a simple structure, and they require a modest amount of memory and computational resources [37]. Consequently, they are a good choice to adopt in some specific application scenarios, such as those characterised by computationally-limited devices and online optimisation processes with hardware-in-the-loop. In this context, a considerable number of single solution algorithms have been proposed. For instance, the online optimisation of an IIR filter and a control system for a micro-helicopter have been addressed respectively in [37] and [10], while in [74] a single solution algorithm has been used to optimise online a neural network for a digit recognition task. Other works [65,35] have proposed single solution algorithms that are capable to obtain similar performances to those offered by the most complex population-based methods on a broad range of benchmark problems. Another class of memory-saving algorithms comes from an adjacent paradigm known as *compact optimisation* [52], i.e., a family of Estimation of Distribution Algorithms that instead of using an explicit population conduct the search by means of a model of the statistical distribution of the solutions, in which the design variables are usually treated as independent. In the following, we will refer to single solution and compact optimisation algorithms collectively as *lightweight metaheuristics*. We will discuss this paradigm more in detail in the next section.

Even though lightweight metaheuristics have been shown to offer good optimisation performances, most of the techniques of this kind suffer from one main drawback: they inherently handle the problem as if it is separable, see e.g. [74,65]. The *separability* property of an optimisation problem indicates that the optimum of that problem can be found by perturbing each variable independently, i.e., the design variables are uncorrelated. However, real-world applications are often characterised by non-separable fitness functions, where variables are correlated and cannot be optimised independently. Some authors refine this distinction even further, by categorising fully separable, moderately separable, moderately non-separable and fully non-separable problems [43], depending on the number of interacting variables. While fully separable problems are usually deemed relatively easy to handle, because each variable can be optimised independently, fully non-separable problems are known to be harder to solve. To properly deal with those problems, in fact, one would normally need to use simultaneous perturbations of multiple variables. However, this kind of perturbations usually require a higher level of algorithmic complexity (for instance, by using a covariance matrix, as in CMA-ES [22]), which on the other hand must be kept low in lightweight metaheuristics. Quite surprisingly though, recent evidence [7] has shown that even non-separable functions can be handled by perturbing separately each variable at a time: while this approach does not necessarily lead to the detection of the optimum, it may still be able to detect promising areas of the decision space. In the same context, another study has shown that the correlation between pairs of variables appears to consistently decrease when the problem dimensionality increases [8]. In other words, non-separable problems in high dimensionalities can be tackled *as if they are separable*. The authors of that study conjectured that this effect is due to the fact that in high dimensionalities only a very restricted portion of the decision space can be explored. So, the best strategy to make use of the available budget is to exploit any improvement along each variable.

It is quite clear then that the way variables are perturbed, i.e., the *mutation* strategy adopted during the search process, is crucial to the algorithmic performance of lightweight metaheuristics. Previous works have investigated the effect of several mutation techniques in the context of population-based algorithms for continuous optimisation, such as Differential Evolution [58,34,67,66] and Genetic Algorithms [60]. For instance, in the field of Differential Evolution mutations acting on random splits of the problem [58], ensemble of mutation strategies [34,67], and the effect of mutation strategies on the convergence rate have been investigated. Concerning Genetic Algorithms instead, in [60] it has been shown with an extensive experimental campaign that a special kind of mutation operator, named Non-Uniform Mutation (NUM) [46], performs better than random mutation (RM), polynomial mutation (PLM) [15], and power mutation (PM) [16]. To the best of our knowledge, no previous work has studied thoroughly the effect of the NUM operator in lightweight metaheuristics.

Based on the aforementioned evidence on the effect of separable operators on non-separable problems [7,8], and being motivated by the results concerning the Non-Uniform Mutation [60] in Genetic Algorithms, in the present work we aim to address the following research questions:

1. Since the original Non-Uniform Mutation (NUM) operator [46] perturbs multiple variables at a time, would it be possible to simplify it by perturbing just a *single* variable at a time (thus treating the problem as if it is separable), to preserve the algorithmic simplicity of lightweight metaheuristics?

2. Would this “Single” Non-Uniform Mutation operator (in the following, SNUM) still lead to an advantage also when dealing with non-separable problems?
3. Would SNUM be useful in both single solution and compact optimisation settings?

To answer these questions, we firstly design a simple single solution algorithm that makes use of SNUM. This scheme is characterised by an iterative perturbation of one variable of the current best solution at a time. Secondly, leveraging the simple design of SNUM, we apply the same principle to a novel compact Genetic Algorithm variant, that we dub cSNUM, to verify how the SNUM operator can be applied to different lightweight algorithmic schemes. In order to validate the proposed SNUM and cSNUM algorithms, we then test them on two well-known benchmarks [21,2], analyse the effect of their parameterisation, and compare their performance against six state-of-the-art lightweight metaheuristics as well as six advanced population-based metaheuristics.

The rest of the paper is organised as follows. In Section 2, we discuss some works related to the scope of the two proposed algorithms. Section 3 defines the perturbation mechanism and the algorithmic design of the proposed SNUM-based methods. Section 4 describes the experimental setup and presents the numerical results. Finally, Section 5 concludes the paper.

2 Lightweight metaheuristics: single solution and compact optimisation algorithms

As discussed in the previous section, in this work we consider two kinds of lightweight metaheuristics, namely *single solution* and *compact optimisation* algorithms. While these two groups of algorithms are rather different in terms of working logic (based, respectively, on a single solution or a probabilistic model), they have one important feature in common: they are *memory-saving* [28]. As shown in previous research, such as e.g. in [28,7,9], this makes them appropriate when facing the need to obtain satisfactory optimisation performance with much less memory than population-based metaheuristics. Such scenarios are not uncommon in real-world problems, where the difficulty aggravates especially if one confronts very large-scale problems [18] or embedded systems such as wireless sensors [29]. In the following, we briefly summarise some of the main related works belonging to the two categories of lightweight metaheuristics.

2.1 Single solution algorithms

While several flavours of single solution algorithms exist, here we will mainly focus on three kinds: 1) Simulated Annealing (and variants thereof); 2) Single Particle Optimisation (and its variants); and 3) Memetic Computing approaches.

Simulated Annealing (SA) [40] is a well-established optimisation technique inspired by the annealing process applied in metallurgy. The main feature of SA is that, differently from greedy approaches (which update the current solution only in case of improvements), it also accepts (i.e., update the current solution to) worse solutions, with a dynamic probability decreases during the optimisation process based on a quantity called “temperature”. A recent variant of SA is the non-uniform Simulated

Annealing (nuSA) [65], which employs the same Non-Uniform Mutation (NUM) operator we use in this work (see Section 3.1) for generating a new candidate solution. As we will discuss below, the NUM operator controls dynamically the range of the search space of the algorithm by progressively reducing the neighbourhood size of the current solution, see Figure 1. This adaptive neighbourhood allows the algorithm to start with an exploratory behavior and become more exploitative as the search process advances, thus overcoming the poorer exploration/exploitation balance of the original SA.

As an alternative to SA algorithms, in the past couple of decades several Single Particle Optimisation methods have been proposed, i.e., versions of Particle Swarm Optimisation (PSO) [39] that make use of a single particle instead of a whole swarm. One instance of this kind of algorithms is the Simplified Intelligence Single Particle Optimisation (ISPO) [74]. ISPO uses a custom rule to update the velocity vector of the particle, based on a learning factor, which is then used to perturb the position of the particle within the search space. The main limitation of ISPO is that, being based on a single particle only, it has an intrinsic exploitation pressure, which is not properly counterbalanced by exploration. In fact, its performance tends to deteriorate on highly multimodal problems [74]. Additionally, its parameters are heavily problem-dependent and finding a proper setting is far from being a trivial task. In order to overcome these limitations and to improve upon the performance of ISPO, ISPO-Restart and VISPO were proposed in [35]. ISPO-Restart combines the logic of ISPO with a partial restart mechanism similar to the binomial crossover typically used in Differential Evolution, while VISPO builds on top of the restart process a very simple learning stage which tries to adapt the algorithm behaviour to the non-separability of the problem. The numerical results reported in [35] have shown that VISPO performs especially well at large scale problems (100D to 1000D), resulting on par with complex population-based algorithms such as CLPSO [44] and JADE [71].

Finally, it is worth mentioning some simple yet efficient single solution Memetic Computing approaches proposed in [10,37,7,9], which are characterised by an iterative application of multiple search strategies. Here, we use the notion of "Memetic Computing" introduced in [50], that is the study of algorithms composed of multiple logical units (memes, i.e., search operators) which are properly coordinated in order to tackle an optimisation problem. More specifically, the algorithms proposed in [10,37, 7] are characterised by a purely sequential structure, composed of a cascade of two or more search units activated sequentially, each one perturbing a single solution according to a different logic (i.e., either a series of stochastic perturbations acting at local or global level, or a series of deterministic local search steps). On the contrary, the Parallel Memetic Structures introduced in [9] are characterised by multiple search units (based on the same perturbation logics used in [10,37,7]) which are executed in a mutually exclusive way: compared to the sequential structures, these parallel structures allow a more flexible execution of the search units and thus an adaptive search process. All the single solutions algorithms introduced in [10,37,7,9] have been proven to outperform several complex population-based algorithms on a broad set of optimisation problems at different dimensionalities.

2.2 Compact optimisation algorithms

As for compact optimisation, one of the earliest algorithms of this kind devised for continuous optimisation was the real-valued compact Genetic Algorithm (rcGA) [47], which in turn extended the seminal works by Harik et al. [26,25] on binary compact Genetic Algorithms. While rcGA was originally proposed for embedded microcontroller optimisation, recently it has been applied to solve even very large-scale problems on a GPU [18]. Over the years, more real-valued compact optimisation algorithms have been developed. The first variant of compact Differential Evolution algorithm (cDE) was presented in [48]. cDE employs the fundamental DE logic for generating new trial solutions but instead of selecting solutions from a population, it samples them from a probabilistic model. The combination of the mutation scheme, the crossover operator (binomial/exponential) and the elitism scheme has an impact on the performance of cDE and leads to a variety of cDE configurations. Besides that, many other variants of cDE have been proposed, see [53,51,32,38,57], among others. The performance of some of them was also investigated in real-world applications, see e.g. [33] where cDE obtained competitive performances on the online tuning of a PID controller for a mobile robot.

A follow-up of these works focused on embodying several Swarm Intelligence (SI) methods into a compact structure, highlighting the fact that while compact optimisation algorithms are normally thought as Evolutionary Algorithms (EAs), compact SI algorithms can also been reinterpreted in the form of an EA. For some SI metaheuristics, the compact version was obtained straightforwardly. As for others, specific modifications and adaptations in terms of formulas and/or steps of their population-based counterparts were required. In this direction, several compact SI algorithms have been proposed, the first one being the compact Particle Swarm Optimisation (cPSO) [54]. This algorithm benefits from the same selection scheme (one-to-one spawning) employed in PSO (as well as in DE) and additionally handles the best solution ever detected (called *elite* in the compact optimisation jargon) as the global best typical of PSO. Following cPSO, a plethora of other compact optimisation algorithms have been proposed, inspired by different kinds of swarm behaviour [36,69,70,11,3,63,12,14, 72]. Other works focused on applications, e.g. in the context of Wireless Sensor Networks [13], hydroelectric power generation [62], or 3D path planning of underwater unmanned submersibles [59].

3 Proposed algorithms

A clear distinction between the two proposed algorithms is that SNUM is a single solution algorithm, whilst cSNUM is a compact Genetic Algorithm variant. In this section, we describe both algorithms and particularly how the Non-Uniform Mutation operator adaptively adjusts the step size of the mutation process.

3.1 Non-Uniform Mutation operator

As observed in [46] in the context of Evolutionary algorithms, keeping the magnitude of random mutations fixed during the evolutionary process may lead to an inefficient search: in fact, while larger mutations may be needed at the beginning to favour exploration, smaller mutations may be useful at the later stages of the optimisation to favour exploitation and thus refine the search. In order to obtain this adaptive behaviour, the Non-Uniform Mutation (NUM) operator has been proposed in [46], and later applied e.g. in [73,65]. This operator is defined as follows. Let us indicate with $\mathbf{x}^{it} = \{x_1, \dots, x_k, \dots, x_D\} \in R^D$ (being D the problem dimensionality) a real-coded vector encoding the current solution at the it^{th} iteration (being it the iteration counter), and with x_k its element selected randomly for variation through this mutation. The resulting offspring solution produced by NUM will be $\mathbf{x}^{it+1} = \{x_1, \dots, x'_k, \dots, x_D\}$, where:

$$x'_k = \begin{cases} x_k + \Delta(it,\ U_k - x_k), \text{ if } \eta = 1 \\ x_k - \Delta(it,\ x_k - L_k), \text{ if } \eta = -1 \end{cases} \tag{1}$$

with η being a random digit that takes either the value -1 or 1, and L_k and U_k being the lower and upper bounds of the variable x_k respectively. The function $\Delta(it, y)$ returns a value in the range $[0, y]$ such that $\Delta(it,\ U_k - x_k)$ approaches zero as the iteration number it increases. To compute this function, the following equation is adopted:

$$\Delta(it, y) = y \times (1 - \rho^{(1-(it/MaxIt))^B}) \tag{2}$$

where: ρ is a uniform random number generated from $\mathcal{U}(0,1)$, $MaxIt$ is the maximum number of iterations, and B is a parameter determining the degree of dependency on the iteration number (non-uniformity). Eq. (2) depends on the iteration counter it, i.e., as it increases the perturbation behaves differently. As analysed in [73], this mutation is neither a Gaussian mutation, which searches only locally (around its mean value), nor a Cauchy mutation, which takes longer steps in the decision space. On the contrary, its characteristics change over time accordingly to the phase of the optimisation process. Figure 1 illustrates how the step size (radius of the neighborhood) changes over the iterations when applying the NUM operator. As it can be seen, the operation spans the whole space uniformly with long jumps at the early stages of the optimisation process. With the progress of the search, the length of the jumps decreases. This makes the search explore very locally smaller and smaller regions at later stages.

3.2 Single Non-Uniform Mutation (SNUM) algorithm

The key idea of our first algorithm is to exploit the above-described NUM operator inside a simple single solution scheme. At the beginning of the optimisation process, a solution is initialised with a random point in the search space, the *elite*. Thereafter, at each iteration of the algorithm, an offspring is created by first copying the current best solution (the "elite"), and then applying on it the NUM operator. In particular, the NUM operator is applied to a *single* decision variable, chosen randomly (hence, the name *Single* Non-Uniform Mutation). Next, the offspring is evaluated and in case of improvement the elite is updated. The latter is finally passed to the next iteration of the algorithm, and the process continues until a stop criterion is verified. The pseudocode of the proposed SNUM algorithm is given in Algorithm 1, where the $SNUM()$ function indicates the application of the NUM operator described in Section 3.1 on a single variable, randomly selected from a candidate solution $\mathbf{x}_{off}$, and the $compete()$ function simply returns the best solution between the two solutions given as arguments.

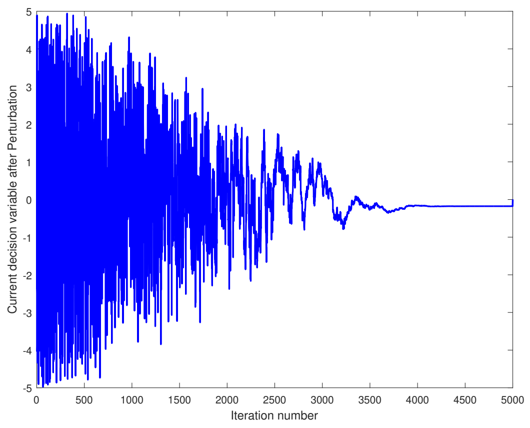


Fig. 1 Behaviour of the Non-Uniform Mutation operator on a 1D sphere function with boundary $[-5,5]$. The vertical axis shows the value of the current decision variable after applying the operator, while the horizontal axis shows the iteration number.

Because the algorithm is based on just one simple operator, SNUM, its behaviour inherits the same core principle of this operator. In other words, the algorithm explores a wide neighbourhood almost covering the whole search space of the current optimal solution at early stages, and a very local neighbourhood at later stages, as shown in Figure 1.

Algorithm 1 Pseudo-code of the proposed SNUM algorithm.

```
Input: Problem bounds, fitness function
Output: elite
Parameters: $B$, $MaxIt$
Initialise a solution within the problem bounds: elite
while $it < MaxIt$ do
    $\mathbf{x}_{off} \leftarrow$ elite
    $\mathbf{x}_{off} \leftarrow SNUM(\mathbf{x}_{off}, B, MaxIt)$        ▷ NUM on a single variable, randomly selected
    elite $\leftarrow compete(\mathbf{x}_{off}, \mathbf{elite})$
    it ← it+1
end while
```

3.3 Compact Single Non-Uniform Mutation (cSNUM) algorithm

The SNUM-based single solution algorithm presented above can be straightforwardly encoded into the real-valued compact Genetic Algorithm (rcGA). We dub this second proposal as cSNUM. Similarly to rcGA, cSNUM uses D independent Gaussian Probability Distribution Functions (PDFs), truncated in the range $[-1,1]$, each one characterised by a mean value μ_i and a standard deviation σ_i. The PDFs are normalised such that their area in $[-1,1]$ is unitary.

For the sake of clarity, Algorithm 2 shows the pseudo-code of the proposed cSNUM. At first, a $2 \times D$ matrix, namely the *perturbation vector* $\mathbf{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$, is initialised such that the $\boldsymbol{\mu}$ values are set equal to zero and the $\boldsymbol{\sigma}$ values are set equal to a large number

$\lambda = 10$. The value of λ is empirically set in order to simulate a uniform distribution at the beginning of the optimisation process. An initial solution **elite** is then sampled from **PV**. For further details on the sampling mechanism, the reader is referred to [47]. Subsequently, at each iteration a new solution $\mathbf{x}_{off}$ is sampled from **PV**, on which the SNUM operator is applied in the same way as in the SNUM algorithm. The fitness of the mutated solution is computed and compared (see the method $compete()$ in the pseudo-code) to the fitness of the current **elite**. On the basis of their fitness values, a **winner** solution and a **loser** solution are then detected. The **winner** solution biases the **PV** values, according to two updating rules [47] which in turn depend on a parameter N_P, that is the "virtual" population size. Finally, the **elite** is updated, according to a persistent elitism scheme, see [1] for details.

Algorithm 2 Pseudo-code of the proposed compact SNUM algorithm (cSNUM).

```
Input: $\mathbf{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}], D$
Output: elite
Parameters: $N_P$, $B$, $MaxIt$
for $i = 1 : D$ do
    Initialise $\mu_i \leftarrow 0$
    Initialise $\sigma_i \leftarrow \lambda = 10$
end for
Generate elite by means of $PV$
while $it < MaxIt$ do
    Generate a solution $\mathbf{x}_{off}$ by means of PV
    $\mathbf{x}_{off} \leftarrow SNUM(\mathbf{x}_{off}, B, MaxIt)$
    $[\mathbf{winner}, \mathbf{loser}] \leftarrow compete(\mathbf{x}_{off}, \mathbf{elite})$
    for $i = 1 : D$ do
        $\mu_i^{it+1} \leftarrow \mu_i^{it} + \frac{1}{N_P}(winner_i - loser_i)$
        $\sigma_i^{it+1} \leftarrow \sqrt{(\sigma_i^{it})^2 + (\mu_i^{it})^2 - (\mu_i^{it+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)}$
    end for
    if $\mathbf{x}_{off}$ == winner then
        elite $\leftarrow \mathbf{x}_{off}$
    end if
    it ← it+1
end while
```

4 Experimental results

In the following, firstly we describe our experimental setup, i.e., the benchmark problems we opted for to assess the performance of the proposed SNUM and cSNUM algorithms, and the algorithms used as comparison basis (Section 4.1), which include other well-established lightweight metaheuristics, i.e., both single solution and compact optimisation algorithms. Then, we present the parameter analysis of SNUM and cSNUM (Section 4.2), followed by a detailed view of the results (Sections 4.3-4.4). Then, we show the results obtained in a separate set of experiments, where we compare the two proposed methods against modern population-based metaheuristics that took part in the CEC-2017 competition (Section 4.5). Finally, we conclude our discussion with the analysis of the time and space complexity of the two proposed methods (Section 4.6).

4.1 Experimental setup: benchmark problems and compared algorithms

We use two well-known sets of test problems, namely the BBOB [21] and the CEC-2017 [2] benchmarks. The former consists of 24 functions, in two dimensionalities: D = 20 and 40; the latter contains 30 test functions in four dimensionalities: D = 10, 30, 50 and 100. Thus, on the whole, we consider 54 benchmark functions.

The following algorithms, with their parameter setting suggested in the original papers, are the basis of our comparative analysis:

– Real-valued compact Genetic Algorithm (rcGA) [47], with persistent elitism, and virtual population size $N_P = 300$.
– Compact Differential Evolution (cDE) [48], with persistent elitism, $rand/1$ mutation and binomial crossover. As for the parameter setting, the cDE mutation is applied with virtual population size $N_P = 300$, scale factor $F = 0.5$ and crossover rate $CR = 0.3$.
– Non-Uniform Simulated Annealing (nuSA) [65], with mutation constant $B = 5$ (note that this has the same meaning of B in Eq. (2)), temperature reduction ratio (also called annealing ratio) $\alpha = 0.9$, temperature reduction period (also called Markov chain length) $M_n = 3$, and number of initial solutions to set the initial temperature equal to 10. Starting from this initial value, the temperature decreases every M_n function evaluations by a factor α. As the temperature is decreased, the probability of accepting a large decrease in solution quality decays exponentially to zero, according to the Boltzmann distribution.
– Simplified Intelligent Single Particle Optimisation (ISPO) [74] with acceleration $A = 1$, acceleration power factor $P = 10$, learning coefficient $B = 2$, learning factor reduction ratio $S = 4$, minimum threshold on learning factor $E = 1e-5$, and particle learning steps $PartLoop = 30$;
– Two improved variants of ISPO: ISPO-Restart (ISPOR) and VISPO [35]. The same parameter setting of ISPO is chosen also for ISPOR, except that A is initialised with respect to the search space bounds ($A = Ub - Lb$) and the restart threshold is set to 0.01. As for VISPO, it is executed with a learning period $LP = 10$, number of perturbations $H = 30$ and learning threshold 0.65.

Thus, we compare the two proposed approaches against six other algorithms in total. For each test function of the BBOB benchmark, each algorithm is executed over 15 runs, as recommended in [21]. The COCO platform[1] is used to ensure the coherence of the results and to obtain a consistent statistical analysis. As for the CEC-2017 functions, each algorithm is executed in 51 independent runs on each problem. Every single run is executed for a fixed budget of $5000 \times D$ fitness evaluations. All the experiments are conducted on a machine with an I7 2.4 GHz CPU and 16 GB of memory. As for the software configuration, we use Matlab 2018 on a Windows 8 operating system. It is also worth mentioning that a toroidal handling [37] of the search space bounds is used in all algorithms analysed in the study.

4.2 SNUM and cSNUM parameterisation

As for the first algorithm, SNUM, all the numerical experiments have been executed with learning coefficient $B = 10$. This value was chosen after testing the algorithm

[1] `https://coco.gforge.inria.fr`

on the entire BBOB benchmark functions in 20 and 40 dimensions, setting B to all the values in the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 60, 70, 80, 90, 100, 200\}$. The detailed results of these experiments are shown in Figures 5 and 6 reported in Appendix A. Overall, we have found that almost all values give similar performances, which means another advantage of the SNUM algorithm: it is not very sensitive to its parameter values, in addition to the advantage of having only one main parameter, that is B.

As for cSNUM, the parameters $B = 8$ and $N_P = 1$ have been chosen after a series of preliminary tuning experiments. More specifically, in order to tune N_P, we fixed B to 6 as an initial guess, and we decreased the N_P value starting from the one used in most compact optimisation algorithms from the literature, i.e, 300. We tested all the N_P values in the set $\{300, 200, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1\}$. Also in this case, we performed the parameter tuning on all the BBOB functions in 20 and 40 dimensions. Overall, we observed that the performance of cSNUM improves as N_P decreases, see Figures 7 and 8 reported in Appendix A for details. In this regard, it is worth mentioning that the effect of N_P is to control the convergence of compact optimisation algorithms. In principle, a lower value is expected to cause premature convergence (with reference to the update rules of μ and σ shown in Algorithm 2, lower N_P value yield larger update steps, which might prevent smaller refinements of **PV**). However, in our case we observed that the lowest value, $N_P = 1$, gave the best performance. According to our interpretation, the exploration/exploitation balance that in rcGA is controlled by using a larger N_P value, in cSNUM is controlled by the SNUM operator itself, which allows to use $N_P = 1$ without compromising the optimisation efficiency. After setting N_P to 1, we then tried to vary the B value in the set $\{1,2,3,4,5,6,7,8,9,10,20,30,40,60,70,80,90,100,200\}$, testing the algorithm again on all the BBOB functions in $20D$ and $40D$. The results of this tuning experiment are displayed in Figures 9 and 10 reported in Appendix A. The final value chosen for the comparison with the other algorithms was then $B = 8$.

4.3 Results on the BBOB functions

The main measure of performances used here is the Expected Running Time (ERT), as described in [20]. This metric depends on a given target function value, $f_t = f_{opt} + \Delta f$, where f_{opt} is the optimal function value and Δf is the precision to reach. The ERT is computed over all the runs as the number of function evaluations needed to reach f_t during each run, summed over all runs and divided by the number of runs that actually reached f_t [21].

From the comparison to the state-of-the-art algorithms[2], expressed in the form of Empirical Cumulative Distribution Function (ECDF) graphs, see Figures 2 and 3 for 20 and 40 dimensions respectively, it can be noted that cSNUM shows a robust performance on both dimensionalities when all functions are taken as a whole, compared to all the other tested algorithms. Furthermore, cSNUM seems to perform better than SNUM, which in turn obtains similar performances to those given by VISPO on all functions. Also, compared to algorithms employing a similar compact logic, the gap of performances in favor of cSNUM over rcGA and cDE increases as the dimensionality

[2] Complete numerical results are available at:
`https://drive.google.com/drive/folders/1yYm3Q4sYpp2kx-lMTPu-8SH-8_RLPAU-?usp=sharing`

increases. Besides this, one can observe that rcGA displays, by far, the worst performance, as expected: as it was already shown in [48], rcGA suffers from premature convergence for dimensionality values higher than 10. This observation further highlights the merits of the SNUM operator: in fact, cSNUM is basically an rcGA with an extra operator, yet this modification appears to be dramatically beneficial in terms of performance.

When each group of functions is considered separately, SNUM displays the best performances on the separable functions for both 20 and 40 dimensions, being notably the only algorithm that solved all functions in this category in all runs. Its performances is statistically significantly superior to those of rcGA, cDE, nuSA, ISPO and ISPOR, and marginally better than cSNUM and VISPO.

Furthermore, we can observe that the moderate functions are handled better by cSNUM. Its superiority is remarkably better compared to SNUM and all other rivals for both 20 and 40 dimensions.

On another note, we can also see that both cSNUM and SNUM, as well as all the other algorithms, do not seem to address the multimodal functions (f15-f19) as effectively as nuSA, for both 20 and 40 dimensions. In addition, although nuSA is in general not among the top algorithms on the whole benchmark, it appears clearly superior on the ill-conditioned functions (f10-f14) for $D = 20$, followed by cSNUM, see Figure 2(d). However, the performance of nuSA deteriorates when dimensionality increases to 40, in which case it is outperformed by most algorithms including ours, as shown in Figure 3(d).

Finally, on the weakly structured multimodal functions, either for $D = 20$ or $D = 40$, cSNUM ranks first in comparison with all the other algorithms.

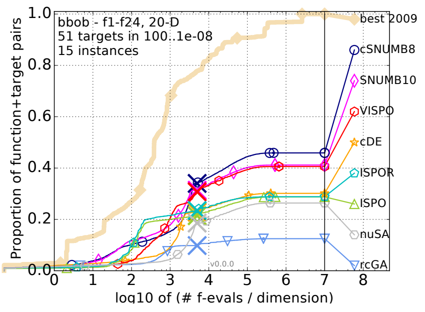
To conclude the analysis on the BBOB benchmark, the main observation is that, even though perturbing one variable at a time in general does not lead to the detection of the optimum, this simple strategy still makes the cSNUM algorithm able to detect promising areas of the decision space, also on non-separable functions.

4.4 Results on the CEC-2017 functions

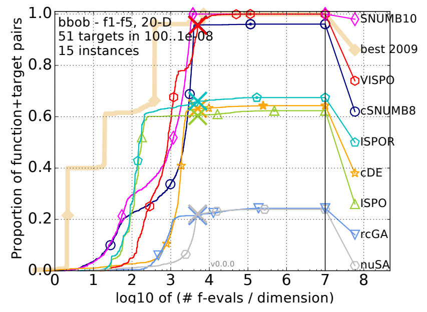
The CEC-2017 benchmark [2] has 3 unimodal functions, 7 simple multimodal functions, 10 hybrid functions and 10 composition functions, which make up a total of 30 functions. All functions are non-separable. To evaluate the performance of the proposed cSNUM and SNUM algorithms on this benchmark and strengthen the interpretation of the numerical results, we use two statistic tests, namely the Wilcoxon signed-rank test and the Holm-Bonferroni procedure, which are non-parametric tests particularly suitable for comparing evolutionary and swarm intelligence algorithms [17].

4.4.1 Statistical comparison of the tested algorithms by means of the Wilcoxon signed-rank test

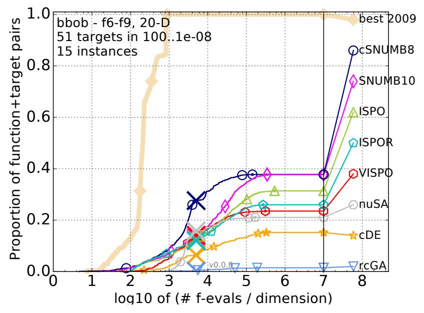
Tables 8-11 (shown in Appendix B) report the results obtained when applying all algorithms under comparison, ours included, on the CEC-2017 benchmark. Data are shown in terms of average fitness error and its standard deviation. Furthermore, for each test problem, we perform a pairwise comparison between one algorithm taken as reference (in each table: cSNUM) and the other algorithms, by applying the Wilcoxon



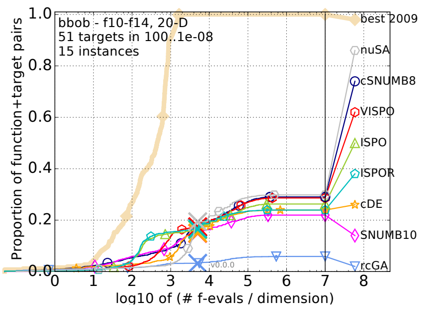
(a) All functions: f1-f24



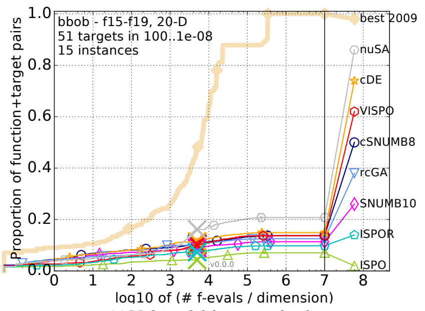
(b) Separable functions: f1-f5



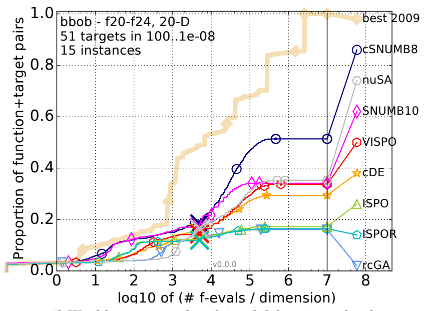
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14



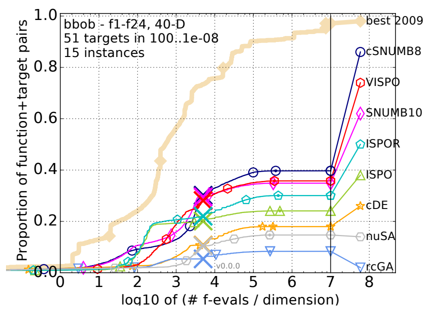
(e) Multimodal functions: f15-f19



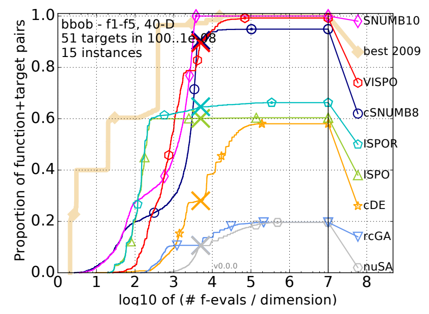
(f) Weakly-structured multimodal functions: f20-f24

Fig. 2 Comparing cSNUM and SNUM to other lightweights algorithms in 20D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.

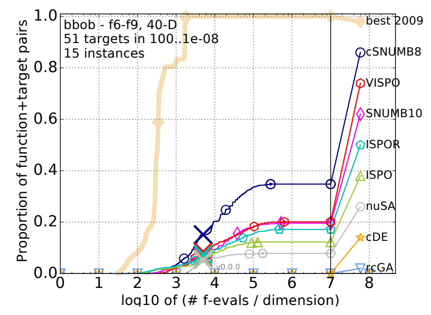
signed-rank test [64], with a confidence level of 0.95 ($\alpha = 0.05$). In the tables, the boldface indicates the algorithm that obtain the minimum average error on each tested benchmark function. The symbol "+" indicates that the reference algorithm cSNUM statistically outperforms the competitor, "-" indicates that cSNUM is outperformed, and "=" is shown when the performance of the two algorithms is statistically equivalent according to the Wilcoxon test.



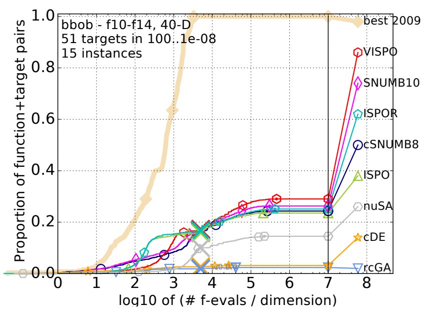
(a) All functions: f1-f24



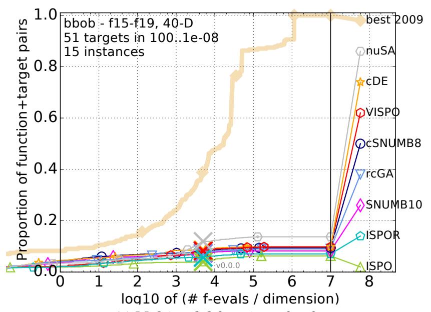
(b) Separable functions: f1-f5



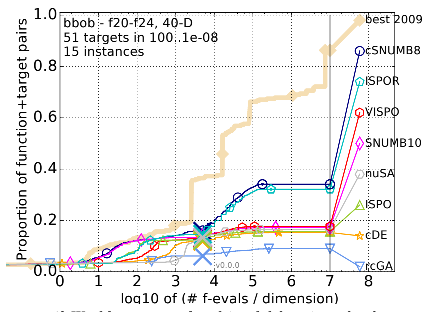
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14



(e) Multimodal functions: f15-f19



(f) Weakly-structured multimodal functions: f20-f24

Fig. 3 Comparing cSNUM and SNUM to other lightweights algorithms in 40D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality $(FEvals/D)$ for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.

For the sake of clarity, a compact representation of the main experimental results is reported in Table 1. Based on the obtained numerical results, if we compare the number of wins (+) and losses (-) obtained by cSNUM taking into account each dimensionality separately, we can see that cSNUM outperforms all algorithms in 10D, with the exception of cDE and nuSA where cSNUM displayed almost similar performance as both of them. When the dimensionality increases to 30, the advantage of cSNUM compared to all the other algorithms slightly increases, apart from nuSA that again exhibits a similar performance. When 50 and 100 dimensions are considered, a clear superiority of cSNUM can be perceived, compared to all the other algorithms, including nuSA and SNUM. This is in accordance with what already pointed out in [8]. That is, insofar as a high-dimensional non-separable problem is tackled as separable and the use of simultaneous variations of multiple variables is avoided, this will enhance the efficiency of the search. The correlation between pairs of variables decreases when the problem dimensionality increases due to the fact that only a very restricted portion of the decision space can be explored in high dimensionalities. Because of this "localness" of the search, the most effective approach to make use of the available budget is to exploit any improvement along each variable.

When we consider all dimensionalities, cSNUM gives better/worse results in 78/6 cases out of 120, 100/9 out of 120, 71/21 out of 120, 65/43 out of 120, 95/7 out of 120, 89/10 out of 120 and 68/16 out of 120, compared to SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO, respectively.

Conclusively, notwithstanding that our results show that nuSA appears to be a good competitor for cSNUM at dealing with non-separable functions, especially in low dimensionalities (and in this sense it should be noted that also nuSA is based on Non-Uniform Mutation), overall cSNUM displays the best performance and seems to be preferable, in all dimensionalities, to all the other algorithms considered in this study.

Table 1 Number of statistically significant losses (-), ties (=) and wins (+) of cSNUM against SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO on the CEC-2017 benchmark [2] in 10, 30, 50 and 100 dimensions.

Optimiser	10D (-/=/+)	30D (-/=/+)	50D (-/=/+)	100D (-/=/+)	TOT (-/=/+)
SNUM	2/9/19	1/6/23	1/14/15	2/7/21	6/36/78
rcGA	6/10/14	2/1/27	1/0/29	0/0/30	9/11/100
cDE	10/12/8	7/10/13	4/6/20	0/0/30	21/28/71
nuSA	12/5/13	13/4/13	10/1/19	8/2/20	43/12/65
ISPO	2/4/24	3/4/23	1/5/24	1/5/24	7/18/95
ISPOR	2/5/23	4/4/22	2/7/21	2/5/23	10/21/89
VISPO	7/10/13	4/8/18	4/9/17	1/9/20	16/36/68
TOT (-/=/+)	41/55/114	34/37/139	23/42/145	14/23/168	112/162/566

4.4.2 Statistical ranking of the tested algorithms by means of the Holm-Bonferroni procedure

In addition to the analysis presented above, we compute the ranking among all the 8 algorithms over the 30 CEC-2017 problems in four dimensionalities (thus for a total of 120 problems) by means of the Holm-Bonferroni procedure [27,19]. The Holm-Bonferroni procedure consists of the following. Considering the results in Tables 8-11, the 8 algorithms under analysis are ranked on the basis of their average performance calculated over the 120 test problems. More specifically, a score R_i for $i = 1,\ldots,N_A$ (where N_A is the number of algorithms under analysis, $N_A = 8$ in our case) is assigned. The score is assigned in the following way: for each problem, a score of 8 is assigned to the algorithm displaying the best performance, 7 is assigned to the second best, 6 to the third and so on. The algorithm displaying the worst performance scores 1. For each algorithm, the scores obtained on each problem are averaged over the amount of test problems (120 in our case). On the basis of these scores, the algorithms are then ranked. With the calculated R_i ranks, cSNUM is taken as a reference algorithm. Indicating with R_0 the rank of cSNUM, and with R_j for $j = 1,\ldots,N_A$-1 the rank of the remaining 7 algorithms, the values z_j are calculated as:

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}} \tag{3}$$

where N_{TP} is the number of test problems in consideration (N_{TP} = 120 in our case). By means of the z_j values, the corresponding cumulative normal distribution values p_j are calculated. These p_j values are then compared with the corresponding δ/j where δ is the level of confidence, set to 0.05 in our case. Table 2 displays the z_j, p_j, and corresponding δ/j values for all the algorithms under analysis. The values of z_j and p_j are expressed in terms of $z_{N_A} - j$ and $p_{N_A} - j$ for $j = 1,\ldots,N_A - 1$. The rank of cSNUM is shown in the caption of the table. Moreover, the table indicates whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected", i.e., cSNUM statistically outperforms the algorithm under consideration, or "Not rejected", i.e., there is no outperformance. As shown in the table, the proposed cSNUM ranks first amongst all the eight algorithms considered in this study. The next best algorithm is nuSA, which is in line with our previous observations. VISPO, SNUM and cDE display respectable performances, which confirms what we noted earlier on the BBOB functions where SNUM and VISPO show similar performances. One can also observe that ISPOR is superior to ISPO on the CEC-2017 benchmark, which proves the usefulness of the restart strategy. As for rcGA, it is ranked the last in the list, similarly to what we found on the BBOB functions.

Table 2 Holm-Bonferroni procedure (reference: cSNUM, Rank = 6.43E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions on all dimensionalities. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	nuSA	5.55E+00	-2.79E+00	2.61E-03	5.00E-02	Rejected
2	VISPO	4.89E+00	-4.88E+00	5.44E-07	2.50E-02	Rejected
3	SNUM	4.82E+00	-5.11E+00	1.59E-07	1.67E-02	Rejected
4	cDE	4.78E+00	-5.24E+00	7.85E-08	1.25E-02	Rejected
5	ISPOR	4.15E+00	-7.22E+00	2.59E-13	1.00E-02	Rejected
6	ISPO	3.02E+00	-1.08E+01	1.64E-27	8.33E-03	Rejected
7	rcGA	2.38E+00	-1.28E+01	5.32E-38	7.14E-03	Rejected

4.5 cSNUM and SNUM vs modern population-based metaheuristics

This comparison is conducted using the results, directly extracted from the original raw data, of six modern population-based algorithms that participated in the CEC

2017 competition on single objective optimisation. This collection of algorithms is representative of a broad set of metaheuristics ranging from evolutionary computation, in particular modern variants of Differential Evolution (JSO [6], LSHADE_SPACMA [49]) and Evolution Strategies (RB_IPOP_CMAES [5]), to swarm intelligence, namely the most modern variants of the Effective Butterfly Optimiser (EBOwithCMAR [42]), the Teaching Learning Based Optimisation algorithm (TLBO_FL [41]), and the Proactive Particle Swarm Optimisation (PPSO [61]). This comparison is intended to demonstrate how two straightforward, lightweight metaheuristics like SNUM and cSNUM perform compared to more complex optimisers that use sophisticated search logics.

All algorithms are tested for D = 10, 30, 50, and 100 dimensions on the 30 CEC-2017 benchmark functions. Table 3 shows the outcome of the Holm-Bonferroni procedure, while Tables 12-19 (shown in Appendix C and D respectively for the comparison with cSNUM and SNUM) include the detailed results in terms of the average error $\pm$ standard deviation together with the outcome of the Wilcoxon signed-rank test. Finally, Table 4 contains a summary of the obtained results.

Table 3 Holm-Bonferroni procedure (Reference: LSHADE_SPACMA , Rank = 6.75E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions on all dimensionalities. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	6.07E+00	-2.16E+00	1.54E-02	5.00E-02	Rejected
2	EBOwithCMAR	5.91E+00	-2.66E+00	3.89E-03	2.50E-02	Rejected
3	JSO	5.30E+00	-4.59E+00	2.27E-06	1.67E-02	Rejected
4	PPSO	3.74E+00	-9.51E+00	9.25E-22	1.25E-02	Rejected
5	cSNUM	3.38E+00	-1.07E+01	6.83E-27	1.00E-02	Rejected
6	TLBO_FL	3.16E+00	-1.14E+01	3.39E-30	8.33E-03	Rejected
7	SNUM	2.08E+00	-1.48E+01	1.38E-49	7.14E-03	Rejected

Table 4 Number of statistically significant losses (-), ties (=) and wins (+) of (cSNUM, SNUM) against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO and TLBO_FL on the CEC-2017 benchmark in 10, 30, 50, and 100 dimensions.

Optimiser	10D (-/=/+)	30D (-/=/+)	50D (-/=/+)	100D (-/=/+)	TOT (-/=/+)
EBOwithCMAR	(29/1/0, 29/0/1)	(26/2/2, 28/0/2)	(25/2/3, 25/3/2)	(16/5/9, 19/3/8)	(96/10/14, 101/6/13)
JSO	(24/2/4, 26/2/2)	(23/1/6, 24/2/4)	(20/3/7, 23/1/6)	(13/6/11, 21/1/8)	(80/12/28, 94/6/20)
LSHADE_SPACMA	(29/1/0, 30/0/0)	(28/1/1, 28/0/2)	(25/1/4, 26/1/3)	(22/2/6, 23/3/4)	(104/5/11, 107/4/9)
RB_IPOP_CMA_ES	(26/1/3, 27/0/3)	(27/3/0, 29/0/1)	(28/0/2, 28/0/2)	(62/2/2, 27/1/2)	(107/6/7, 111/1/8)
PPSO	(19/3/8, 23/3/4)	(17/4/9, 22/4/4)	(12/8/10, 18/5/7)	(11/1/18, 16/1/13)	(59/16/45, 79/13/28)
TLBO_FL	(17/4/9, 22/3/5)	(15/5/10, 19/5/6)	(12/7/11, 16/7/7)	(8/1/21, 11/1/18)	(52/17/51, 68/16/36)
TOT (-/=/+)	(144/12/24, 157/8/15)	(136/16/28, 150/11/19)	(122/21/37, 136/17/27)	(96/17/67, 117/10/53)	(498/66/156, 560/46/114)

When all functions in all dimensionalities are considered together, cSNUM and SNUM do not achieve overall competitive results and are not able to outperform these algorithms in the majority of test problems. This is evident in Table 3, where the reference algorithm is LSHADE_SPACMA, which has the highest rank and SNUM is at the bottom of the table which has the lowest rank. Given that cSNUM ranks before TLBO_FL algorithm, scrutinising cases where cSNUM (or SNUM) excels may provide a way to understand its behaviour.

In Table 5, we indicate the functions where cSNUM (or SNUM) outperforms the advanced population-based algorithms, based on the mean result. It can be seen that cSNUM (or SNUM) can still find better solutions than other high-quality algorithms, including the highest performer ones (EBOwithCMAR, LSHADE_SPACMA, JSO, and RB_IPOP_CMA_ES).

Table 5 CEC 2017 benchmark functions on which cSNUM (or SNUM) outperforms EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO and TLBO_FL in 10, 30, 50, and 100 dimensions.

	D10	D30	D50	D100
All	({∅} / {∅})	({∅} / {∅})	({∅} / {∅})	({6} / {6})
EBOwithCMAR	({∅} / {∅})	({2,6,10}/ {10})	({6,10,20} / {6,10})	({4,6,10,16,17,20,22-24} / {6,10,16,17,20,22-24})
JSO	({6,7,10,20} / {10,20})	({5-8,10,21,23} / {5-7,10})	({5-8,10,21-23} / {5-8,10,21,22})	({4-8,10,16,17,20,22-24,29} / {6,7,10,16,17,20,22,23})
LSHADE_SPACMA	({∅} / {25})	({2,10} / {10})	({6,10,21,23} / {10,21,23})	({4,6,10,21,23,24} / {6,10,21,23,24})
RB_IPOP_CMA_ES	({6,10,17,20} / {6,17,20})	({2,6} / {∅})	({11,12,30} / {12,30})	({6,11,30} / {6,30})
PPSO	({2,3,5,6,10,11,20,23,27} / {6,20,23})	({2,5,6,7,10,11,23,24,27,28} / {6,23,24,27})	({4,6,7,9,10,21-23,25-29} / {6,7,23,25-28)	(1,4,6,7,11,12,18,21-30} / {1,4,6,7,12,21,23-28,30})
TLBO_FL	({2,3,4,5,7,8,10,12,20,30} / {7,10,20,25})	({2,3,4,6,7,10,11,13,15,25,28,30} / {4,6,7,10,15,25,28,30})	({1-4,6,7,10,11,18,20,22,25,27,28} / {1,2,4,6,7,10,22,25,27,28,30})	({1-4,6,7,10-14,18,20,22-28,30} / {1,2,4,6,7,10,12-14,18,20,22-28,30})

Based on the results summarised in Tables 4-5, one can notice that both cSNUM and SNUM achieve good results when the dimensionality increases, in particular to $D = 100$. To confirm this, we apply the Bonferroni procedure on each dimensionality apart. Tables 20-23 (shown in Appendix E) correspond to 10, 30, 50 and 100 dimensions, respectively. Another finding is that cSNUM ranks before PPSO and TLBO_FL algorithms for $D = 100$. This proves that cSNUM works better at large dimensionalities.

4.6 Algorithmic complexity

Our experimental campaign concludes with an analysis of the time complexity and the memory requirements of the proposed algorithms.

4.6.1 Time complexity

We first measure the time complexity using the procedure defined in the COCO platform [21]. According to this procedure, the overall CPU time is determined by running the algorithm on the BBOB $f8$ function (Rosenbrock) for a given number of iterations. The CPU time is recorded for each dimensionality per function evaluation. The COCO developers recommend running the time complexity experiment in the same dimensionalities as the benchmarking experiment; however, here we expand the test by adding more dimensionalities (10, 20, 30, 40, 50, 100, 300, 500, 700, 800 and 1000) to get a better approximation of the complexity related to the problem dimensionality. As depicted in Figure 4, the proposed cSNUM and SNUM have both linear time complexity.

Second, we calculate the time complexity based on the indications of the CEC-2017 benchmark. Table 6 summarises the algorithmic complexity of cSNUM and SNUM, in terms of average computational time, on $D = 10$, 30, 50, and 100, respectively. In the table, $\hat{TC}$ is the average time (across 30 independent runs per each algorithm) to execute the proposed algorithms with the same parameters described above and a budget of 200,000 $FEvals$ on the $f18$ function from CEC-2017 in each dimensionality. Again, from both columns of the table, one can notice that the average CPU time of both algorithms increases linearly when the dimensionality increases. Conclusively, the time

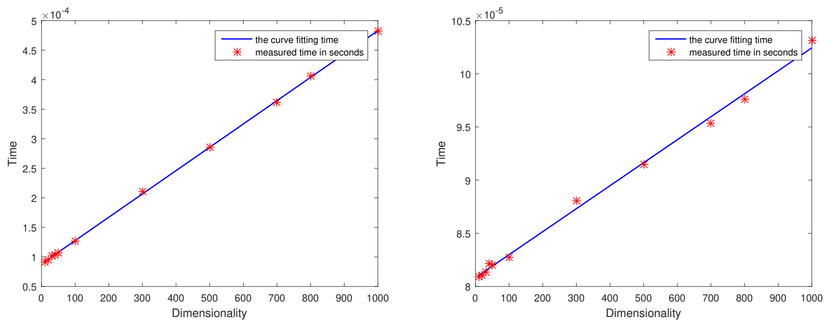


Fig. 4 Time [s] needed to run cSNUM (left) and SNUM (right) on the BBOB $f8$ function at different dimensionalities.

complexity of both cSNUM and SNUM does not significantly increase when the problem dimensionality increases. This highlights the reduced time complexity of the two proposed algorithms, which could be used as effective optimisation tools to identify acceptable solutions for higher-dimensional problems in scenarios characterised by limited computing resources.

Table 6 Time [s] needed to run cSNUM and SNUM on the CEC-2017 $f18$ function at different dimensionalities.

Dimensionality	$\hat{TC}$[s] (cSNUM)	$\hat{TC}$[s] (SNUM)
10	3.02	1.02
30	4.79	1.33
50	6.62	1.76
100	12.49	4.04

Third, we estimate the theoretical complexity, in terms of $\mathcal{O}$ notation, of the two proposed algorithms. As seen in Algorithm 1, the proposed SNUM repeats $MaxIt$ times a set of three instructions, two of which having a $\mathcal{O}(1)$ complexity each (the first and third), and one (the second instruction) with $\mathcal{O}(D)$ complexity, due to the sampling of $\mathbf{x}_{off}$ from a probabilistic distribution. As seen in Algorithm 2, we observe instead that cSNUM is composed of two main blocks, namely:

– an initialisation block: it repeats two basic instruction about D times, so its complexity is $\mathcal{O}(D)$.
– a second block, repeated $MaxIt$ times, composed of: a call to the "Generate a solution" function, with complexity $\mathcal{O}(D)$; a call to the SNUM operator, with complexity $\mathcal{O}(D)$; a for loop with D repetitions, thus with $\mathcal{O}(D)$ complexity; in addition to other instructions with complexity $\mathcal{O}(1)$.

Thus, on the whole, both SNUM and cSNUM have $\simeq \mathcal{O}(D)$ time complexity, which is in line with the empirically measured execution times shown in Figure 4 and Table 6, although it is to note that cSNUM is relatively slower than SNUM.

Finally, one has to bear in mind that this complexity is computed for the case where $MaxIt$ does not depend on the problem dimensionality. Otherwise, a quadratic complexity would appear for both algorithms.

4.6.2 Space complexity

The second complexity aspect we consider in our analysis is memory consumption. This is an especially relevant feature for lightweight algorithms such as the proposed cSNUM and SNUM algorithm, which clearly distinguish them from population-based metaheuristics. In order to assess space complexity, we compare the 14 algorithms considered in our experimentation (the 2 proposals, 6 lightweight algorithms, and 6 modern metaheuristics) in terms of the approximate number of D-dimensional vectors used to perform the algorithmic operations in memory. The information contained in Table 7 reveals that single solution algorithms including ISPO, ISPOR, VISPO, nuSA and SNUM require slightly less memory than compact optimisation algorithms such as rcGA, cDE and cSNUM. In addition, the table clearly shows, as expected, that compared to population-based algorithms (EBOwithCMAR, LSHADE_SPACMA, JSO, RB-IPOP-CMAES, PPSO and TLBO_FL) all lightweight algorithms have considerably lower memory requirements.

Table 7 Memory complexity in terms of approximate number of "memory slots" (i.e., D-dimensional vectors) of the algorithms analysed in the experimentation.

Algorithm class	Algorithm	Number of memory slots
Compact	rcGA	4
	cDE	4
	cSNUM	**4**
Single solution	nuSA	2
	ISPO	2
	ISPOR	2
	VISPO	2
	SNUM	**2**
Population-based	EBOwithCMAR	$PopSize = PS_1 + PS_2 + PS_3$+HistoricalMemory ($PS_1$ = 64.8*D, PS_2 = 4, PS_3= 3*log(D), HistoricalMemory= 6)
	LSHADE_SPACMA	$PopSize$+ MemorySize ($PopSize = 18 * D$, $MemorySize$ = 5)
	JSO	2*$PopSize$+HistoricalMemory ($PopSize$=25*$log(D)\sqrt{D}$, HistoricalMemory= 5)
	RB-IPOP-CMAES	2+⌊(3/2)*log(D)⌋+(nbrGenerations-1)*(16+⌊12*log(D)⌋)
	PPSO	3*$PopSize$+1 ($PopSize$=200/300/500, according to the dimensionality D)
	TLBO_FL	$PopSize$+1 ($PopSize$ =100)

5 Conclusion

This paper illustrates the value of the Single Non-Uniform Mutation (SNUM) operator, by investigating its impact when employed into two schemes characterised by a different logic. The first SNUM-based proposal (also called SNUM) has a notably simple and efficient structure for single solution progressive perturbation. The second approach (cSNUM) shows how the SNUM operator could be easily reused in a compact Genetic Algorithm scheme while preserving its merits. Of note, the two proposed algorithms have modest memory requirements (respectively, they need to keep in memory only 2 and 4 D-dimensional vectors, being D the problem dimensionality) and linear time complexity $\mathcal{O}(D)$, thus being suitable for embedded implementations.

The experiments conducted on a set of 54 reference benchmark functions, with different dimensionalities, show that cSNUM is able to deal with problems at different dimensionalities, outperforming all the other lightweight algorithms under study. Furthermore, cSNUM appears effective at tackling not only separable functions, but also other categories of problems. As for the single solution SNUM algorithm, it shows similar performances to those of another lightweight algorithm, i.e., the VISPO algorithm. However, SNUM by design is characterised by an exceptional minimalist and straightforward algorithmic structure that contains just one operator and only one parameter. Among the compared algorithms, we can observe that nuSA (which is also based on Non-Uniform Mutation) is also a good choice in handling the non-separable problems from the CEC-2017 benchmark in lower dimensionalities, and the multimodal and ill-conditioned functions from the BBOB benchmark in 20D. Furthermore, cSNUM shows the best overall performances and the highest robustness across the different dimensionalities.

Additionally, the statistical analysis of the comparison with two recent DE variants (JSO and LSHADE_SPACMA), a recent variant of CMA-ES (RB_IPOP_CMAES), and three modern swarm intelligence methods (EBOwithCMAR, PPSO and TLBO_FL), prove that cSNUM achieves notable success over some approaches, especially PPSO and TLBO_FL in 100 dimensions.

For further studies, a promising direction would be to integrate the SNUM operator with other algorithmic components that could play somewhat different but complementary roles, in order to deal with non-separable functions even more effectively. In addition, restart mechanisms such as the re-sampled inheritance introduced in [10,30,31] could be considered to further improve performances and avoid premature convergence on highly multimodal problems.

Declarations

Funding: Not applicable.
Conflicts of interest/Competing interests: Not applicable.
Availability of data and material: The raw data are available upon request.
Code availability: The code is available upon request.
Authors' contributions: All authors contributed to the design of this study. Experiments and data collection were performed by S. Khalfi and A. Draa. The analysis was performed by all authors. The first draft of the manuscript was written by S. Khalfi. All authors read and approved the final manuscript.

Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

References

1. Ahn, C.W., Ramakrishna, R.S.: Elitism-based compact genetic algorithms. IEEE Transactions on Evolutionary Computation **7**(4), 367–385 (2003)

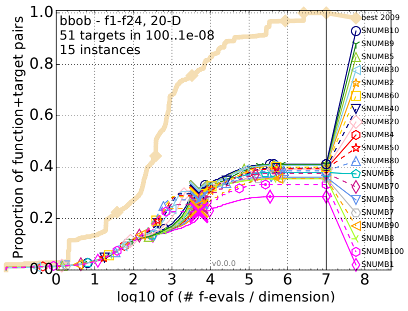
2. Awad N, H., Ali M, Z., Qu B, Y., Liang J, J., Suganthan P, N.: Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Tech. rep., Nanyang Technological University, Singapore (November 2016)
3. Banitalebi, A., Aziz, M.I.A., Bahar, A., Aziz, Z.A.: Enhanced compact artificial bee colony. Information Sciences **298**, 491–511 (2015)
4. Bansal, J.C., Singh, P.K., Pal, N.R.: Evolutionary and swarm intelligence algorithms. Springer (2019)
5. Biedrzycki, R.: A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1489–1494. IEEE (2017)
6. Brest, J., Maučec, M.S., Bošković, B.: Single objective real-parameter optimization: Algorithm jSO. In: 2017 IEEE congress on evolutionary computation (CEC), pp. 1311–1318. IEEE (2017)
7. Caraffini, F., Iacca, G., Neri, F., Mininno, E.: Three variants of three stage optimal memetic exploration for handling non-separable fitness landscapes. In: 2012 12th UK Workshop on Computational Intelligence (UKCI), pp. 1–8. IEEE (2012)
8. Caraffini, F., Neri, F., Iacca, G.: Large scale problems in practice: the effect of dimensionality on the interaction among variables. In: European Conference on the Applications of Evolutionary Computation, pp. 636–652. Springer (2017)
9. Caraffini, F., Neri, F., Iacca, G., Mol, A.: Parallel memetic structures. Information Sciences **227**, 60–82 (2013)
10. Caraffini, F., Neri, F., Passow, B.N., Iacca, G.: Re-sampled inheritance search: high performance despite the simplicity. Soft Computing **17**(12), 2235–2256 (2013)
11. Dao, T.K., Chu, S.C., Shieh, C.S., Horng, M.F., et al.: Compact artificial bee colony. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 96–105. Springer (2014)
12. Dao, T.K., Pan, J.S., Chu, S.C., Shieh, C.S., et al.: Compact bat algorithm. In: Intelligent Data analysis and its Applications, Volume II, pp. 57–68. Springer (2014)
13. Dao, T.K., Pan, T.S., Nguyen, T.T., Chu, S.C.: A compact artificial bee colony optimization for topology control scheme in wireless sensor networks. Journal of Information Hiding and Multimedia Signal Processing **6**(2), 297–310 (2015)
14. Dao, T.K., Pan, T.S., Nguyen, T.T., Chu, S.C., Pan, J.S.: A compact flower pollination algorithm optimization. In: 2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN), pp. 76–79. IEEE (2016)
15. Deb, K., Goyal, M.: A combined genetic adaptive search (geneas) for engineering design. Computer Science and Informatics **26**, 30–45 (1996)
16. Deep, K., Thakur, M.: A new mutation operator for real coded genetic algorithms. Applied Mathematics and Computation **193**(1), 211–230 (2007)
17. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011)
18. Ferigo, A., Iacca, G.: A GPU-enabled compact genetic algorithm for very large-scale optimization problems. Mathematics **8**(5), 758 (2020)
19. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing **13**(10), 959 (2009)
20. Hansen, N., Auger, A., Brockhoff, D., Tušar, D., Tušar, T.: COCO: performance assessment. arXiv preprint arXiv:1605.03560 (2016)
21. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking: Experimental setup. Tech. rep., Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep (2012)
22. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary computation **11**(1), 1–18 (2003)
23. Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: PSO facing non-separable and ill-conditioned problems. Tech. Rep. RR-6447, INRIA (2008)
24. Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. Applied Soft Computing **11**(8), 5755–5769 (2011)
25. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE transactions on evolutionary computation **3**(4), 287–297 (1999)

26. Harik, G.R., Lobo, F.G., et al.: A parameter-less genetic algorithm. In: GECCO, vol. 99, pp. 258–267 (1999)
27. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics pp. 65–70 (1979)
28. Iacca, G.: Memory-saving optimization algorithms for systems with limited hardware. Ph.D. thesis, University of Jyväskylä (2011)
29. Iacca, G.: Distributed optimization in wireless sensor networks: an island-model framework. Soft Computing **17**(12), 2257–2277 (2013)
30. Iacca, G., Caraffini, F.: Compact optimization algorithms with re-sampled inheritance. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), pp. 523–534. Springer (2019)
31. Iacca, G., Caraffini, F.: Re-sampled inheritance compact optimization. Knowledge-Based Systems **208**, 106416 (2020)
32. Iacca, G., Caraffini, F., Neri, F.: Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead. Journal of Computer Science and Technology **27**(5), 1056–1076 (2012)
33. Iacca, G., Caraffini, F., Neri, F.: Memory-saving memetic computing for path-following mobile robots. Applied Soft Computing **13**(4), 2003–2016 (2013)
34. Iacca, G., Caraffini, F., Neri, F.: Multi-strategy coevolving aging particle optimization. International Journal of Neural Systems **24**(01) (2013)
35. Iacca, G., Caraffini, F., Neri, F., Mininno, E.: Single particle algorithms for continuous optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 1610–1617. IEEE (2013)
36. Iacca, G., Neri, F., Mininno, E.: Compact bacterial foraging optimization. In: Swarm and Evolutionary Computation, pp. 84–92. Springer (2012)
37. Iacca, G., Neri, F., Mininno, E., Ong, Y.S., Lim, M.H.: Ockham's razor in memetic computing: three stage optimal memetic exploration. Information Sciences **188**, 17–43 (2012)
38. Jewajinda, Y.: Covariance matrix compact differential evolution for embedded intelligence. In: 2016 IEEE Region 10 Symposium (TENSYMP), pp. 349–354. IEEE (2016)
39. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
40. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
41. Kommadath, R., Kotecha, P.: Teaching learning based optimization with focused learning and its performance on CEC 2017 functions. In: 2017 IEEE congress on evolutionary computation (CEC), pp. 2397–2403. IEEE (2017)
42. Kumar, A., Misra, R.K., Singh, D.: Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase. In: 2017 IEEE congress on evolutionary computation (CEC), pp. 1835–1842. IEEE (2017)
43. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K., China, H.: Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Tech. Rep. 33, IEEE (2013)
44. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation **10**(3), 281–295 (2006)
45. Mahdavi, S., Shiri, M.E., Rahnamayan, S.: Metaheuristics in large-scale global continues optimization: A survey. Information Sciences **295**, 407–428 (2015)
46. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs. Springer (1996)
47. Mininno, E., Cupertino, F., Naso, D.: Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Transactions on Evolutionary Computation **12**(2), 203–219 (2008)
48. Mininno, E., Neri, F., Cupertino, F., Naso, D.: Compact differential evolution. IEEE Transactions on Evolutionary Computation **15**(1), 32–54 (2011)
49. Mohamed, A.W., Hadi, A.A., Fattouh, A.M., Jambi, K.M.: LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In: 2017 IEEE Congress on evolutionary computation (CEC), pp. 145–152. IEEE (2017)
50. Neri, F., Cotta, C., Moscato, P.: Handbook of memetic algorithms, vol. 379. Springer (2011)
51. Neri, F., Iacca, G., Mininno, E.: Disturbed exploitation compact differential evolution for limited memory optimization problems. Information Sciences **181**(12), 2469–2487 (2011)
52. Neri, F., Iacca, G., Mininno, E.: Compact optimization. In: Handbook of optimization, pp. 337–364. Springer (2013)
53. Neri, F., Mininno, E.: Memetic compact differential evolution for cartesian robot control. IEEE Computational Intelligence Magazine **5**(2), 54–65 (2010)

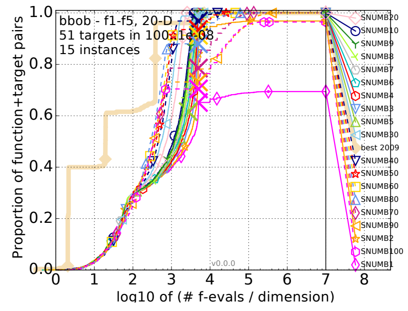
54. Neri, F., Mininno, E., Iacca, G.: Compact particle swarm optimization. Information Sciences **239**, 96–121 (2013)
55. Prügel-Bennett, A.: Benefits of a population: Five mechanisms that advantage population-based algorithms. IEEE Transactions on Evolutionary Computation **14**(4), 500–517 (2010)
56. Rao, S.S.: Engineering optimization: theory and practice. John Wiley & Sons (2019)
57. Sergio, A., Carvalho, S., Marco, R.: On the use of compact approaches in evolution strategies. Advances in Distributed Computing and Artificial Intelligence Journal **3**(4), 13–23 (2014)
58. Shi, Y.j., Teng, H.f., Li, Z.q.: Cooperative co-evolutionary differential evolution for function optimization. In: International Conference on Natural Computation, pp. 1080–1088. Springer (2005)
59. Song, P.C., Pan, J.S., Chu, S.C.: A parallel compact cuckoo search algorithm for three-dimensional path planning. Applied Soft Computing p. 106443 (2020)
60. Tang, P.H., Tseng, M.H.: Adaptive directed mutation for real-coded genetic algorithms. Applied Soft Computing **13**(1), 600–614 (2013)
61. Tangherloni, A., Rundo, L., Nobile, M.S.: Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1940–1947. IEEE (2017)
62. Tian, A.Q., Chu, S.C., Pan, J.S., Cui, H., Zheng, W.M.: A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station. Sustainability **12**(3), 767 (2020)
63. Tighzert, L., Fonlupt, C., Mendil, B.: A set of new compact firefly algorithms. Swarm and Evolutionary Computation **40**, 92–115 (2018)
64. Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in statistics, pp. 196–202. Springer (1992)
65. Xinchao, Z.: Simulated annealing algorithm with adaptive neighborhood. Applied Soft Computing **11**(2), 1827–1836 (2011)
66. Yaman, A., Iacca, G., Caraffini, F.: A comparison of three differential evolution strategies in terms of early convergence with different population sizes. In: International Global Optimization Workshop (2019)
67. Yaman, A., Iacca, G., Coler, M., Fletcher, G., Pechenizkiy, M.: Multi-strategy differential evolution. In: International Conference on the Applications of Evolutionary Computation, pp. 617–633. Springer, Cham (2018)
68. Yang, X.S.: Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons (2010)
69. Yang, Z., Li, K., Guo, Y.: A new compact teaching-learning-based optimization method. In: International Conference on Intelligent Computing, pp. 717–726. Springer (2014)
70. Yang, Z., Li, K., Guo, Y., Ma, H., Zheng, M.: Compact real-valued teaching-learning based optimization with the applications to neural network training. Knowledge-Based Systems **159**, 51–62 (2018)
71. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation **13**(5), 945–958 (2009)
72. Zhao, M., Pan, J.S., Chen, S.T.: Compact cat swarm optimization algorithm. In: International Conference on Security with Intelligent Computing and Big-data Services, pp. 33–43. Springer (2017)
73. Zhao, X., Gao, X.S., Hu, Z.C.: Evolutionary programming based on non-uniform mutation. Applied Mathematics and Computation **192**(1), 1–11 (2007)
74. Zhou, J., Ji, Z., Shen, L.: Simplified intelligence single particle optimization based neural network for digit recognition. In: 2008 Chinese Conference on Pattern Recognition, pp. 1–5. IEEE (2008)

A Parameter analysis of SNUM and cSNUM on BBOB

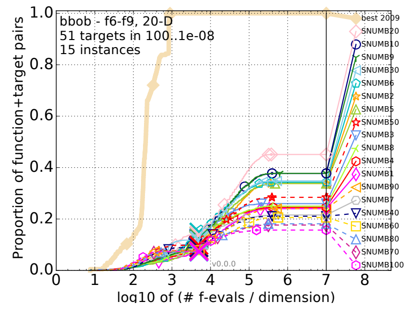
A.1 Effect of the B parameter of SNUM on BBOB



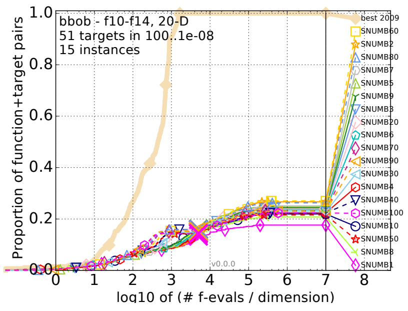
(a) All functions: f1-f24



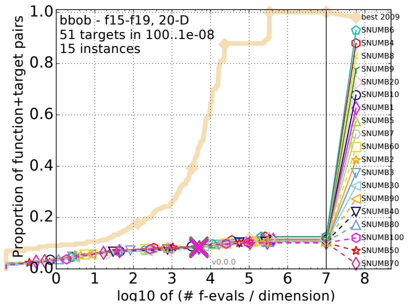
(b) Separable functions: f1-f5



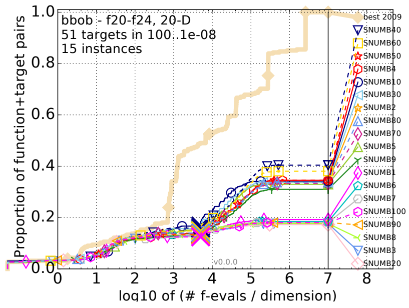
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

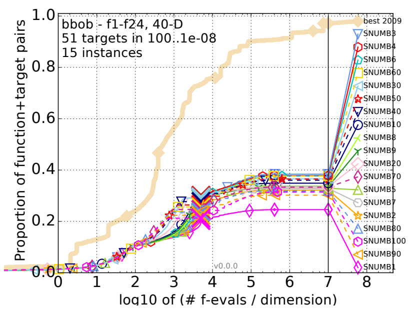


(e) Multimodal functions: f15-f19

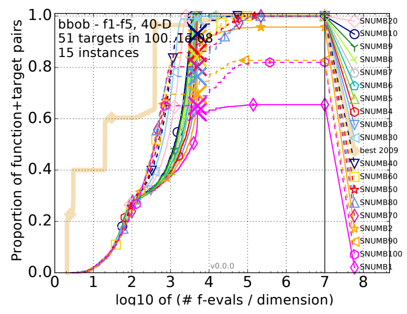


(f) Weakly-structured multimodal functions: f20-f24

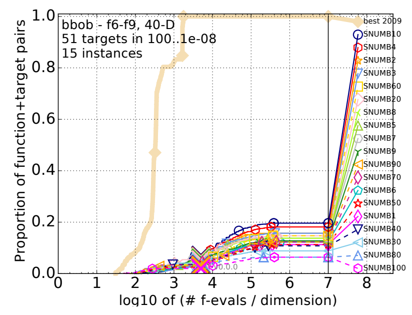
Fig. 5 Effect of the B parameter of SNUM on BBOB in 20D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.



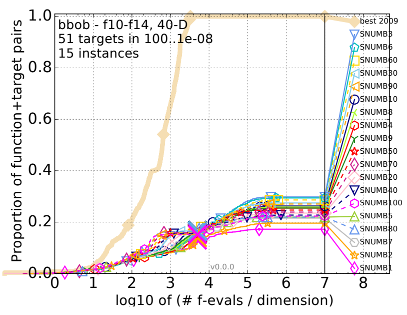
(a) All functions: f1-f24



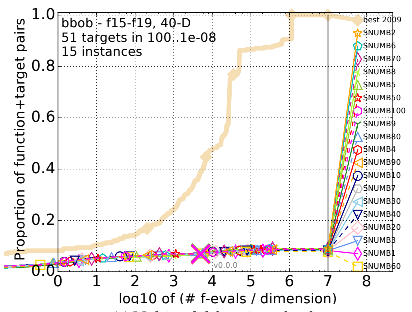
(b) Separable functions: f1-f5



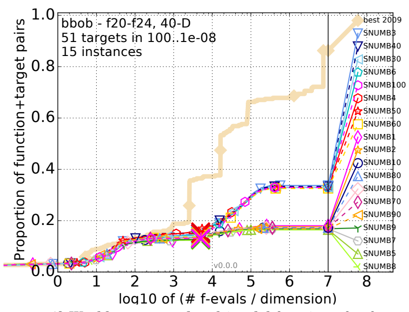
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14



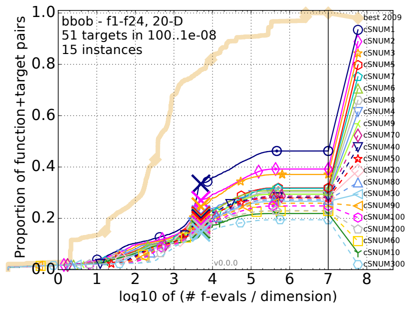
(e) Multimodal functions: f15-f19



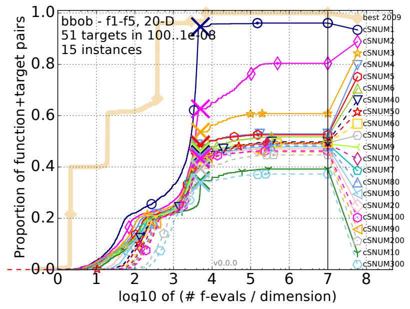
(f) Weakly-structured multimodal functions: f20-f24

Fig. 6 Effect of the B parameter of SNUM on BBOB in 40D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.

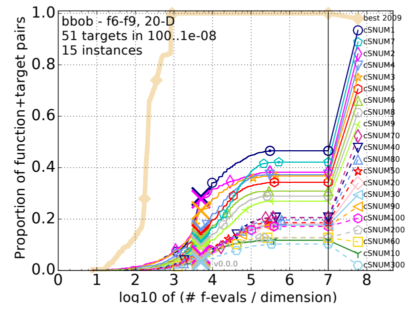
A.2 Effect of the B and N_P parameters of cSNUM on BBOB



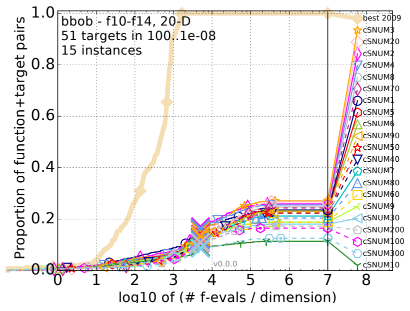
(a) All functions: f1-f24



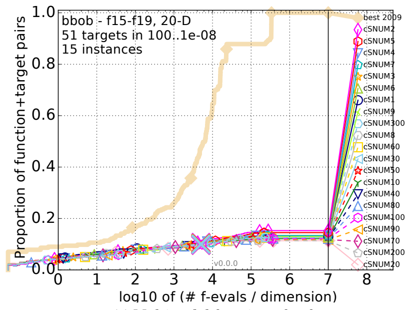
(b) Separable functions: f1-f5



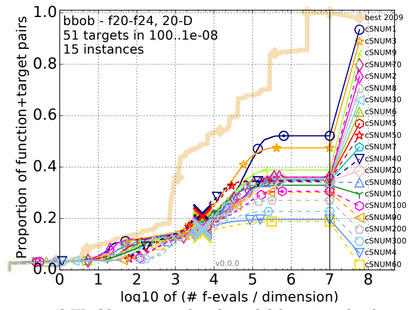
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

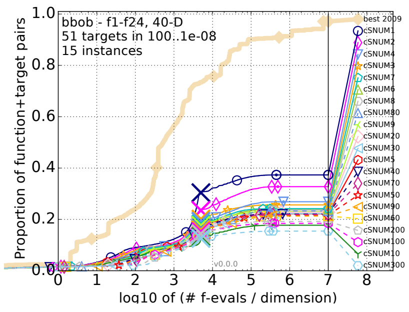


(e) Multimodal functions: f15-f19

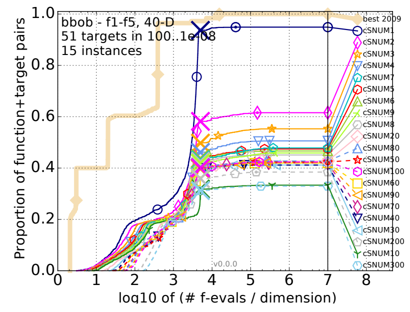


(f) Weakly-structured multimodal functions: f20-f24

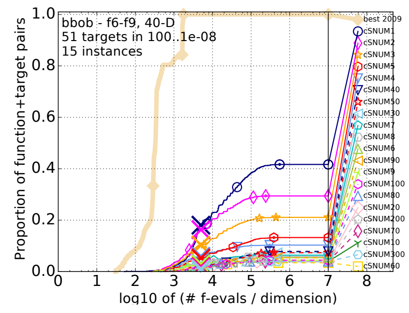
Fig. 7 Effect of the N_P parameter of cSNUM with $B = 6$ on BBOB in 20D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.



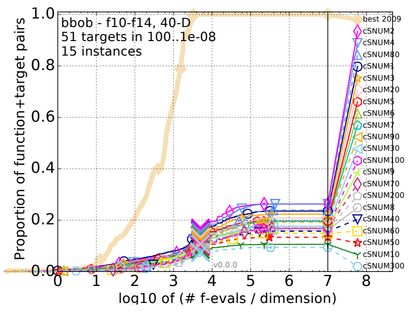
(a) All functions: f1-f24



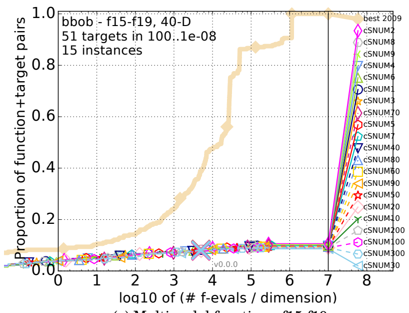
(b) Separable functions: f1-f5



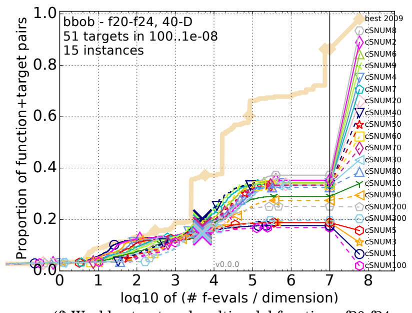
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

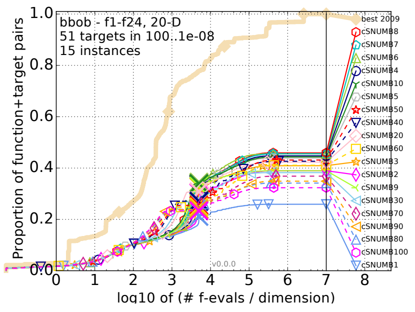


(e) Multimodal functions: f15-f19

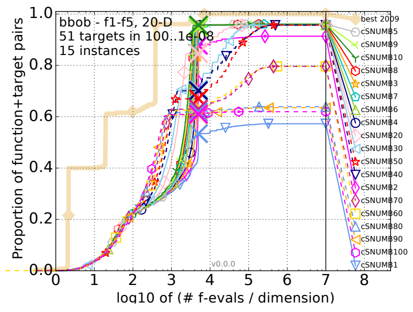


(f) Weakly-structured multimodal functions: f20-f24

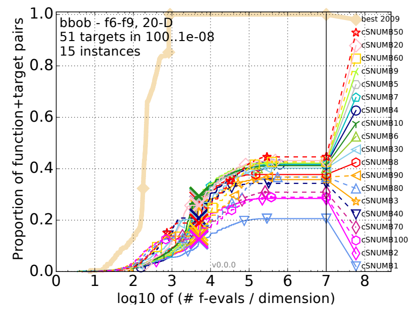
Fig. 8 Effect of the N_P parameter of cSNUM with $B = 6$ on BBOB in 40D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.



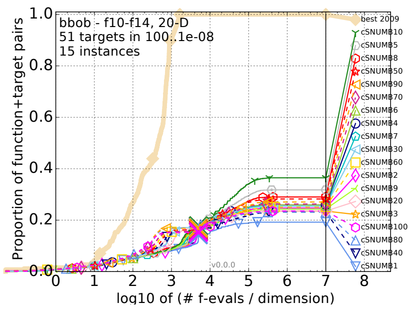
(a) All functions: f1-f24



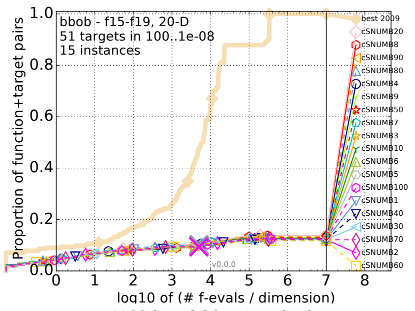
(b) Separable functions: f1-f5



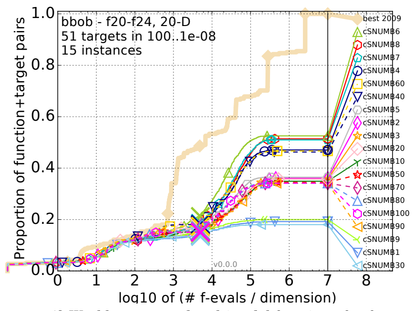
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

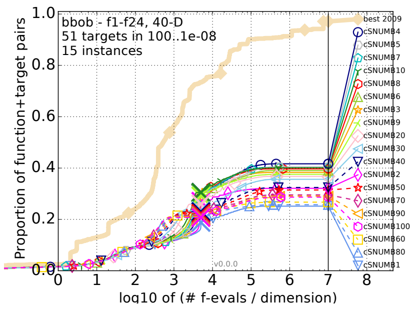


(e) Multimodal functions: f15-f19

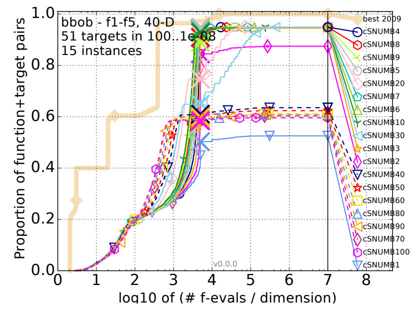


(f) Weakly-structured multimodal functions: f20-f24

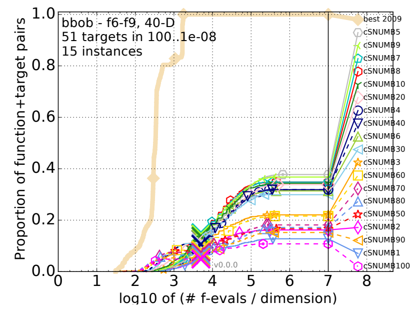
Fig. 9 Effect of the B parameter of cSNUM with $N_P = 1$ on BBOB in 20D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the “best 2009” algorithm from BBOB 2009 is shown as light thick line with diamond markers.



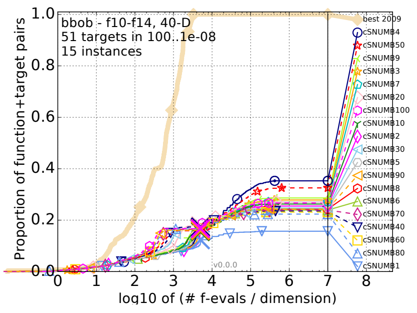
(a) All functions: f1-f24



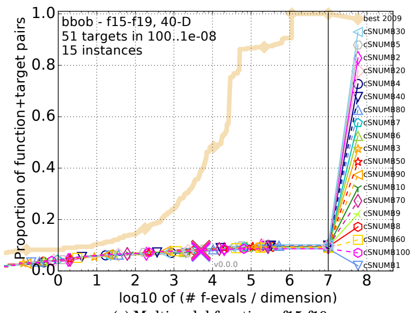
(b) Separable functions: f1-f5



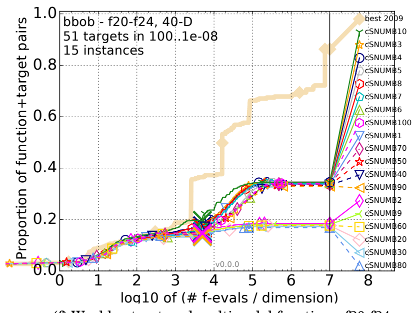
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14



(e) Multimodal functions: f15-f19



(f) Weakly-structured multimodal functions: f20-f24

Fig. 10 Effect of the B parameter of cSNUM with $N_P = 1$ on BBOB in 40D: Bootstrapped Empirical Cumulative Distribution of the number of objective function evaluations divided by problem dimensionality ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.

B Extended numerical results for SNUM and cSNUM on CEC-2017 against lightweight algorithms

Table 8 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO on CEC-2017 [2] in 10 dimensions.

Function	cSNUM		SNUM			rcGA			cDE			nuSA			ISPO			ISPOR			VISPO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	4.93E+03	4.00E+03	5.58E+03	4.36E+03	=	3.20E+06	2.28E+07	=	3.59E+03	3.44E+03	=	**3.40E+03**	**2.94E+03**	-	3.59E+03	2.93E+03	=	3.70E+03	2.82E+03	=	4.65E+04	1.10E+05	+
2	4.16E-04	7.20E-04	1.61E+02	5.94E+02	-	3.99E+07	1.17E+08	+	2.39E+03	4.34E+03	+	1.35E+01	5.82E+00	+	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	9.80E+00	5.32E+01	-
3	**8.19E-13**	**1.45E-12**	4.03E+02	5.87E+02	+	4.30E+03	3.21E+03	+	3.90E-03	2.16E-02	+	2.23E-10	4.16E-10	+	3.70E+03	5.45E+03	+	3.96E+03	5.47E+03	+	5.92E+03	6.12E+03	+
4	3.27E+00	1.38E+01	1.32E+01	2.54E+01	+	2.98E+01	2.80E+01	+	5.16E+00	9.95E+00	+	2.01E+00	5.28E-01	+	**1.33E+00**	**5.81E-01**	+	2.92E+00	1.00E+01	+	7.70E+00	1.57E+01	+
5	1.37E+01	5.07E+00	2.17E+01	8.56E+00	+	2.31E+01	8.05E+00	+	**1.15E+01**	**5.05E+00**	-	1.57E+01	6.57E+00	=	2.41E+02	1.17E+02	+	2.54E+01	1.09E+01	+	1.56E+01	6.85E+00	=
6	**7.49E-07**	**2.05E-06**	6.57E-05	2.34E-04	-	1.28E+01	7.18E+00	+	6.15E-02	8.18E-02	+	1.04E+00	1.02E+00	+	8.26E+01	1.79E+01	+	2.09E+00	1.45E+00	+	7.81E-05	2.37E-04	-
7	**1.81E+01**	**3.76E+00**	3.20E+01	7.82E+00	+	7.12E+01	3.36E+01	+	2.78E+01	7.10E+00	+	2.51E+01	8.05E+00	+	8.81E+02	2.58E+02	+	4.42E+01	1.34E+01	+	3.36E+01	8.19E+00	+
8	1.29E+01	5.85E+00	2.50E+01	1.06E+01	+	3.13E+01	1.10E+01	+	**1.26E+01**	**4.20E+00**	=	1.46E+01	7.99E+00	=	1.48E+02	5.59E+01	+	2.98E+01	1.10E+01	+	1.69E+01	7.75E+00	+
9	3.01E+00	2.15E+01	3.72E+01	1.58E+02	+	3.51E+02	2.66E+02	+	5.63E+00	7.26E+00	+	**1.09E-01**	**4.00E-01**	+	3.20E+03	1.38E+03	+	2.25E+02	3.06E+02	+	5.54E+01	1.60E+02	+
10	4.52E+02	2.14E+02	6.86E+02	2.65E+02	+	8.45E+02	2.92E+02	+	**3.64E+02**	**1.81E+02**	-	5.91E+02	2.60E+02	+	1.86E+03	4.68E+02	+	7.98E+02	2.74E+02	+	4.13E+02	1.72E+02	=
11	**7.12E+00**	**4.52E+00**	1.86E+01	2.67E+01	+	9.95E+01	6.65E+01	+	9.00E+00	4.79E+00	+	1.45E+01	6.98E+00	+	9.29E+01	5.92E+01	+	1.76E+01	9.23E+00	+	2.18E+01	3.22E+01	+
12	1.97E+04	1.56E+04	1.51E+05	2.06E+05	+	2.76E+05	1.21E+06	=	**1.31E+04**	**1.57E+04**	-	5.20E+04	1.69E+04	+	1.28E+05	1.08E+05	+	1.02E+05	9.23E+04	+	3.70E+05	5.41E+05	+
13	**7.51E+03**	**9.02E+03**	7.96E+03	1.08E+04	=	1.14E+04	1.19E+04	+	8.60E+03	9.71E+03	=	9.35E+03	1.03E+04	+	1.76E+04	1.29E+04	+	1.92E+04	1.24E+04	+	1.52E+04	1.12E+04	+
14	8.17E+03	8.55E+03	1.03E+04	9.57E+03	=	6.69E+02	1.94E+03	-	3.50E+03	6.05E+03	-	**6.18E+01**	**2.83E+01**	-	6.31E+03	7.35E+03	=	6.41E+03	7.83E+03	=	2.48E+03	5.02E+03	-
15	6.11E+03	8.54E+03	1.18E+04	1.16E+04	+	3.56E+03	4.43E+03	=	6.33E+03	7.81E+03	=	**3.85E+02**	**4.73E+02**	-	1.01E+04	1.09E+04	+	1.02E+04	1.11E+04	+	7.74E+03	9.38E+03	=
16	1.70E+02	1.39E+02	2.17E+02	1.63E+02	=	9.72E+01	9.25E+01	-	1.06E+02	9.53E+01	-	**4.91E+01**	**6.97E+01**	-	7.83E+02	3.28E+02	+	3.08E+02	1.74E+02	+	1.78E+02	1.37E+02	=
17	5.13E+01	6.38E+01	5.84E+01	7.14E+01	=	7.66E+01	4.88E+01	+	**2.07E+01**	**3.10E+01**	=	4.37E+01	2.51E+01	+	4.09E+02	2.26E+02	+	6.82E+01	6.54E+01	+	4.23E+01	4.86E+01	=
18	1.30E+04	1.22E+04	2.08E+04	1.40E+04	+	1.38E+04	1.01E+04	=	1.21E+04	9.17E+03	=	2.54E+04	1.17E+04	+	1.30E+04	9.55E+03	=	1.31E+04	8.51E+03	=	**1.20E+04**	**1.19E+04**	=
19	1.04E+04	1.13E+04	1.00E+04	1.09E+04	=	8.55E+03	9.77E+03	=	6.78E+03	8.30E+03	=	**1.00E+03**	**1.10E+03**	-	3.21E+03	5.25E+03	-	3.86E+03	7.17E+03	-	4.22E+03	6.85E+03	-
20	**4.27E+00**	**6.90E+00**	1.07E+01	9.14E+00	+	7.48E+01	3.58E+01	+	5.04E+00	5.96E+00	+	3.70E+01	1.53E+01	+	5.76E+02	2.56E+02	+	2.20E+01	1.22E+01	+	7.54E+00	5.99E+00	+
21	2.16E+02	3.03E+01	2.21E+02	4.63E+01	+	**1.09E+02**	**1.73E+01**	-	2.02E+02	4.11E+01	-	1.59E+02	6.28E+01	-	3.69E+02	1.09E+02	+	2.40E+02	5.65E+01	+	1.56E+02	6.44E+01	-
22	2.28E+02	2.96E+02	7.52E+02	5.36E+02	+	1.21E+02	8.53E+01	=	2.09E+02	2.60E+02	=	**8.88E+01**	**2.97E+01**	=	1.73E+03	8.87E+02	+	7.92E+02	6.85E+02	+	2.07E+02	2.72E+02	+
23	3.19E+02	7.67E+00	3.31E+02	2.44E+01	+	3.18E+02	7.02E+00	=	**3.16E+02**	**5.77E+00**	=	3.21E+02	9.90E+00	=	1.04E+03	4.84E+02	+	3.47E+02	1.71E+01	+	3.35E+02	1.92E+01	+
24	3.60E+02	5.23E+01	3.40E+02	8.58E+01	=	**2.74E+02**	**9.54E+01**	-	3.17E+02	8.35E+01	-	3.42E+02	3.56E+01	-	4.76E+02	1.69E+02	+	3.51E+02	9.68E+01	=	3.43E+02	8.31E+01	=
25	4.32E+02	3.14E+01	4.21E+02	6.10E+01	=	4.64E+02	2.11E+01	+	4.29E+02	5.13E+01	=	**4.12E+02**	**2.10E+01**	-	4.62E+02	1.18E+02	+	4.42E+02	5.44E+01	+	4.29E+02	3.03E+01	=
26	8.47E+02	5.45E+02	1.02E+03	5.72E+02	+	4.20E+02	6.39E+01	=	5.20E+02	4.19E+02	-	**3.46E+02**	**2.18E+02**	-	2.26E+03	1.15E+03	+	1.08E+03	6.64E+02	+	5.95E+02	5.35E+02	-
27	4.06E+02	2.33E+01	4.37E+02	3.53E+01	+	3.96E+02	4.32E+00	-	4.01E+02	1.35E+01	=	**3.93E+02**	**2.50E+00**	-	7.19E+02	3.55E+02	+	4.39E+02	3.12E+01	+	4.05E+02	1.50E+01	=
28	5.44E+02	1.44E+02	5.67E+02	1.29E+02	=	4.16E+02	6.03E+01	-	4.38E+02	1.39E+02	-	**3.91E+02**	**1.31E+02**	-	6.10E+02	2.03E+02	=	5.63E+02	1.84E+02	=	4.44E+02	1.21E+02	-
29	3.20E+02	6.19E+01	3.76E+02	8.02E+01	+	3.12E+02	5.49E+01	=	**2.70E+02**	**2.39E+01**	-	2.83E+02	4.38E+01	-	8.58E+02	2.32E+02	+	3.64E+02	7.70E+01	+	2.95E+02	4.63E+01	=
30	2.56E+05	4.19E+05	5.02E+05	6.81E+05	+	1.63E+05	3.63E+05	=	**1.45E+05**	**3.01E+05**	=	4.38E+05	7.00E+05	=	7.99E+05	6.11E+05	+	5.16E+05	5.62E+05	+	4.65E+05	5.83E+05	+

Table 9 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO on CEC-2017 [2] in 30 dimensions.

Function	cSNUM		SNUM			rcGA			cDE			nuSA			ISPO			ISPOR			VISPO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	7.00E+03	6.93E+03	9.53E+03	7.62E+03	+	9.84E+09	4.96E+09	+	1.23E+08	3.41E+08	+	**2.09E+03**	**1.40E+03**	-	3.90E+03	3.97E+03	-	4.06E+03	3.88E+03	-	1.49E+04	6.97E+04	=
2	**1.41E-03**	**2.88E-03**	3.69E+23	2.41E+24	+	7.33E+37	4.00E+38	+	1.46E+27	7.21E+27	+	1.28E+13	1.14E+13	+	3.44E+03	1.24E+04	-	4.71E+03	3.18E+04	-	5.48E+13	3.70E+14	+
3	7.02E+03	7.77E+03	1.04E+05	3.83E+04	+	1.36E+05	2.89E+04	+	4.08E+04	1.55E+04	+	**8.91E+02**	**2.26E+02**	-	1.29E+05	6.09E+04	+	1.17E+05	4.91E+04	+	1.56E+05	5.16E+04	+
4	6.99E+01	2.55E+01	7.37E+01	2.73E+01	+	9.68E+02	8.80E+02	+	1.68E+02	5.66E+01	+	7.52E+01	1.97E+01	=	**5.99E+01**	**3.55E+01**	=	**6.38E+01**	3.63E+01	=	7.22E+01	3.80E+01	=
5	9.09E+01	2.04E+01	1.16E+02	3.45E+01	+	2.23E+02	4.74E+01	+	1.14E+02	2.81E+01	+	**8.31E+01**	**2.00E+01**	-	6.69E+02	1.66E+02	+	1.87E+02	3.94E+01	+	1.14E+02	3.68E+01	+
6	9.69E-07	4.90E-07	6.68E-05	3.34E-04	-	4.31E+01	1.05E+01	+	1.06E+01	3.43E+00	+	1.80E+01	4.66E+00	+	9.29E+01	1.11E+01	+	1.97E+00	6.89E-01	+	**1.40E-08**	**4.26E-09**	-
7	**1.05E+02**	**1.96E+01**	1.36E+02	2.41E+01	+	1.09E+03	3.21E+02	+	2.68E+02	5.45E+01	+	1.36E+02	2.94E+01	+	3.83E+03	7.76E+02	+	2.44E+02	5.19E+01	+	1.77E+02	3.52E+01	+
8	1.11E+02	2.72E+01	1.34E+02	3.45E+01	+	2.29E+02	4.64E+01	+	1.17E+02	2.51E+01	=	**8.01E+01**	**2.10E+01**	-	5.62E+02	1.26E+02	+	2.14E+02	5.75E+01	+	1.18E+02	4.07E+01	=
9	1.77E+03	1.78E+03	2.62E+03	1.93E+03	+	6.96E+03	2.94E+03	+	2.31E+03	1.28E+03	+	**7.04E+02**	**6.61E+02**	-	1.37E+04	3.83E+03	+	6.63E+03	2.82E+03	+	1.49E+03	2.35E+03	-
10	**2.69E+03**	**4.36E+02**	3.22E+03	5.03E+02	+	5.24E+03	8.71E+02	+	3.37E+03	8.02E+02	+	3.69E+03	8.02E+02	+	5.66E+03	8.27E+02	+	3.77E+03	5.43E+02	+	2.97E+03	6.69E+02	+
11	**7.59E+01**	**3.64E+01**	1.15E+02	4.81E+01	+	2.18E+03	2.02E+03	+	1.92E+02	1.99E+02	+	9.67E+01	3.76E+01	+	1.83E+02	5.93E+01	+	1.65E+02	5.43E+01	+	2.67E+02	2.59E+02	+
12	5.75E+05	4.36E+05	1.31E+06	1.09E+06	+	2.94E+08	4.28E+08	+	**5.69E+05**	**1.02E+06**	-	8.07E+05	3.35E+05	+	1.52E+06	9.03E+05	+	1.55E+06	7.67E+05	+	1.78E+06	1.50E+06	+
13	1.95E+04	2.01E+04	2.45E+04	2.55E+04	=	3.36E+07	9.99E+07	+	1.84E+04	2.13E+04	=	5.17E+04	2.79E+04	+	2.55E+04	2.31E+04	=	**1.46E+04**	**1.91E+04**	=	2.69E+04	2.42E+04	=
14	1.18E+05	1.37E+05	9.20E+05	1.07E+06	+	2.72E+05	3.55E+05	+	7.00E+04	7.59E+04	-	**1.04E+04**	**6.65E+03**	-	1.76E+06	1.66E+06	+	1.62E+06	1.64E+06	+	5.11E+05	5.54E+05	+
15	1.62E+04	1.49E+04	2.12E+04	1.46E+04	+	5.43E+04	4.74E+04	+	**1.19E+04**	**1.27E+04**	=	6.07E+04	1.73E+04	+	1.29E+04	1.33E+04	=	1.54E+04	1.36E+04	=	1.57E+04	1.88E+04	=
16	1.27E+03	3.34E+02	1.24E+03	3.53E+02	=	1.46E+03	4.30E+02	+	9.76E+02	2.72E+02	-	**8.09E+02**	**3.10E+02**	-	1.85E+03	5.19E+02	+	1.52E+03	3.93E+02	+	1.43E+03	3.96E+02	+
17	7.14E+02	2.54E+02	7.61E+02	2.49E+02	=	7.25E+02	2.99E+02	=	4.02E+02	1.86E+02	-	**2.77E+02**	**1.43E+02**	-	1.27E+03	4.03E+02	+	8.22E+02	3.08E+02	=	6.64E+02	2.65E+02	=
18	5.49E+05	4.58E+05	3.14E+06	3.50E+06	+	2.44E+06	3.22E+06	+	4.10E+05	3.42E+05	=	**1.74E+05**	**1.20E+05**	-	5.42E+06	4.70E+06	+	4.46E+06	4.59E+06	+	2.42E+06	2.65E+06	+
19	1.36E+04	1.22E+04	1.67E+04	1.74E+04	=	3.79E+06	1.13E+07	+	1.16E+04	1.25E+04	=	7.68E+04	2.67E+04	+	1.02E+04	1.33E+04	-	**9.47E+03**	**1.12E+04**	-	1.21E+04	1.51E+04	=
20	5.84E+02	2.33E+02	7.31E+02	2.96E+02	+	6.92E+02	1.59E+02	+	4.52E+02	1.70E+02	-	**2.99E+02**	**1.26E+02**	-	1.35E+03	3.57E+02	+	8.86E+02	2.62E+02	+	3.79E+02	2.09E+02	-
21	3.18E+02	3.44E+01	3.39E+02	3.55E+01	+	3.96E+02	3.98E+01	+	3.16E+02	3.00E+01	=	**2.87E+02**	**2.30E+01**	-	8.29E+02	1.85E+02	+	4.13E+02	5.28E+01	+	3.42E+02	4.20E+01	+
22	2.59E+03	1.25E+03	3.49E+03	1.04E+03	+	**2.35E+03**	**1.74E+03**	-	3.19E+03	1.13E+03	+	3.73E+03	1.14E+03	+	6.25E+03	1.07E+03	+	4.59E+03	8.40E+02	+	2.62E+03	1.77E+03	+
23	**4.61E+02**	**2.72E+01**	4.78E+02	2.91E+01	+	5.31E+02	3.66E+01	+	4.62E+02	2.48E+01	=	4.70E+02	3.13E+01	=	2.10E+03	7.93E+02	+	5.60E+02	5.92E+01	+	5.02E+02	4.22E+01	+
24	7.04E+02	1.04E+02	7.03E+02	1.11E+02	=	5.82E+02	3.62E+01	-	**5.34E+02**	**2.76E+01**	-	5.48E+02	4.55E+01	-	1.15E+03	2.57E+02	+	8.12E+02	1.10E+02	+	6.69E+02	7.41E+01	=
25	3.93E+02	1.61E+01	3.98E+02	2.12E+01	+	1.34E+03	6.17E+02	+	4.77E+02	4.57E+01	+	4.15E+02	1.67E+01	+	3.98E+02	1.59E+01	+	3.92E+02	1.13E+01	+	**3.89E+02**	**7.62E+00**	-
26	2.28E+03	6.72E+02	2.37E+03	6.26E+02	=	3.37E+03	4.78E+02	+	2.33E+03	4.73E+02	=	**2.06E+03**	**3.10E+02**	-	9.32E+03	2.97E+03	+	3.44E+03	1.10E+03	+	2.69E+03	9.24E+02	+
27	5.35E+02	1.98E+01	5.42E+02	1.75E+01	+	5.72E+02	2.59E+01	+	**5.28E+02**	**1.41E+01**	=	5.94E+02	1.82E+01	+	1.11E+03	7.52E+02	+	5.57E+02	2.24E+01	+	5.42E+02	1.65E+01	+
28	3.79E+02	6.74E+01	4.31E+02	4.04E+01	+	1.13E+03	4.03E+02	+	5.28E+02	5.15E+01	+	3.66E+02	4.62E+01	=	3.62E+02	7.55E+01	=	**3.43E+02**	**6.35E+01**	-	4.00E+02	6.20E+01	+
29	9.79E+02	2.32E+02	1.11E+03	2.58E+02	+	1.42E+03	3.64E+02	+	**8.47E+02**	**2.07E+02**	-	9.95E+02	2.16E+02	=	1.89E+03	3.71E+02	+	1.27E+03	2.65E+02	+	1.24E+03	2.31E+02	+
30	**1.11E+04**	**1.84E+04**	1.60E+04	1.13E+04	+	8.27E+06	2.01E+07	+	1.11E+04	6.82E+03	=	4.29E+05	1.99E+05	+	1.16E+04	5.05E+03	+	1.19E+04	4.95E+03	+	1.28E+04	4.70E+03	+

Table 10 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO on CEC-2017 [2] in 50 dimensions.

Function	cSNUM		SNUM			rcGA			cDE			nuSA			ISPO			ISPOR			VISPO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.42E+04	1.30E+04	1.34E+04	1.25E+04	=	3.94E+10	1.10E+10	+	6.88E+09	3.71E+09	+	2.69E+07	7.94E+06	+	**1.06E+04**	**1.12E+04**	=	1.23E+04	1.13E+04	=	1.14E+05	5.93E+05	=
2	**1.49E+20**	**1.06E+21**	6.62E+35	4.72E+36	+	3.28E+74	2.30E+75	+	7.16E+60	3.94E+61	+	2.92E+29	9.00E+29	+	2.31E+28	8.24E+28	+	2.90E+27	2.04E+28	=	6.11E+36	4.35E+37	+
3	2.09E+04	1.27E+04	2.24E+05	5.44E+04	+	3.04E+05	6.12E+04	+	1.65E+05	3.56E+04	+	**1.29E+04**	**1.51E+03**	-	2.97E+05	7.93E+04	+	3.04E+05	7.58E+04	+	2.87E+05	1.30E+05	+
4	8.46E+01	5.06E+01	1.14E+02	5.90E+01	+	3.93E+03	1.67E+03	+	1.00E+03	3.71E+02	+	1.88E+02	4.38E+01	+	7.84E+01	4.77E+01	=	**7.77E+01**	**4.49E+01**	=	9.69E+01	5.99E+01	=
5	2.12E+02	3.76E+01	2.29E+02	4.90E+01	=	5.50E+02	8.75E+01	+	3.39E+02	4.59E+01	+	**1.74E+02**	**3.37E+01**	-	1.06E+03	2.01E+02	+	3.97E+02	7.44E+01	+	2.44E+02	6.19E+01	+
6	**1.52E-06**	**6.20E-07**	4.23E-05	2.12E-04	-	6.46E+01	1.12E+01	+	2.68E+01	5.13E+00	+	3.06E+01	4.04E+00	+	9.23E+01	7.62E+00	+	1.92E+00	5.34E-01	+	5.47E-05	2.27E-04	-
7	**2.04E+02**	**2.93E+01**	2.58E+02	3.91E+01	+	3.05E+03	4.91E+02	+	1.08E+03	1.47E+02	+	2.77E+02	4.68E+01	+	6.44E+03	1.08E+03	+	4.91E+02	7.77E+01	+	3.70E+02	6.34E+01	+
8	2.11E+02	4.96E+01	2.40E+02	5.43E+01	+	5.28E+02	8.40E+01	+	3.43E+02	4.14E+01	+	**1.77E+02**	**3.22E+01**	-	9.93E+02	2.18E+02	+	3.94E+02	7.84E+01	+	2.27E+02	6.18E+01	=
9	6.09E+03	4.05E+03	6.60E+03	3.91E+03	=	2.49E+04	5.91E+03	+	1.18E+04	4.07E+03	+	5.65E+03	5.88E+03	=	3.06E+04	5.95E+03	+	1.91E+04	5.92E+03	+	**2.37E+03**	**2.43E+03**	-
10	**4.85E+03**	**8.05E+02**	5.25E+03	7.71E+02	+	1.04E+04	1.31E+03	+	7.73E+03	8.34E+02	+	7.03E+03	1.50E+03	+	8.97E+03	1.03E+03	+	6.47E+03	8.53E+02	+	5.63E+03	6.91E+02	+
11	**1.52E+02**	**4.33E+01**	1.62E+03	2.43E+03	+	1.27E+04	5.90E+03	+	2.13E+03	2.38E+03	+	2.15E+02	2.76E+01	+	1.35E+03	1.79E+03	+	1.19E+03	1.63E+03	+	3.84E+03	4.91E+03	+
12	**1.84E+06**	**1.34E+06**	2.62E+06	1.48E+06	+	4.95E+09	2.87E+09	+	3.26E+08	4.34E+08	+	6.78E+06	2.04E+06	+	2.10E+06	1.15E+06	=	2.39E+06	1.51E+06	=	3.31E+06	3.17E+06	+
13	**1.01E+04**	**1.02E+04**	1.65E+04	1.42E+04	+	1.32E+09	1.23E+09	+	2.92E+04	3.66E+04	+	5.27E+04	2.19E+04	+	1.40E+04	1.03E+04	+	1.07E+04	1.04E+04	=	1.28E+04	1.33E+04	=
14	1.92E+05	1.05E+05	7.60E+05	5.23E+05	+	2.36E+06	2.05E+06	+	1.12E+05	8.59E+04	-	**5.38E+04**	**3.69E+04**	-	2.09E+06	1.38E+06	+	2.27E+06	1.54E+06	+	1.62E+06	1.64E+06	+
15	1.16E+04	9.03E+03	1.48E+04	8.95E+03	=	8.70E+07	1.86E+08	+	1.08E+04	9.06E+03	=	4.64E+04	1.22E+04	+	**9.96E+03**	**7.84E+03**	=	1.04E+04	8.35E+03	=	1.84E+04	5.18E+04	=
16	2.13E+03	3.66E+02	2.27E+03	4.95E+02	=	2.89E+03	7.28E+02	+	1.98E+03	4.32E+02	=	**1.45E+03**	**4.82E+02**	-	2.91E+03	5.50E+02	+	2.50E+03	5.45E+02	+	2.31E+03	5.35E+02	=
17	1.62E+03	3.14E+02	1.59E+03	3.82E+02	=	3.02E+03	7.56E+02	+	1.46E+03	3.36E+02	-	**1.08E+03**	**3.01E+02**	-	2.53E+03	4.62E+02	+	1.98E+03	3.74E+02	+	1.79E+03	3.59E+02	+
18	6.88E+05	4.33E+05	2.78E+06	2.04E+06	+	1.04E+07	1.19E+07	+	8.76E+05	9.47E+05	=	**2.65E+05**	**8.38E+04**	-	2.27E+06	1.60E+06	+	3.03E+06	2.47E+06	+	4.47E+06	5.71E+06	+
19	1.68E+04	1.53E+04	1.94E+04	1.70E+04	=	3.30E+07	8.92E+07	+	1.48E+04	1.35E+04	=	1.37E+05	4.08E+04	+	1.16E+04	1.31E+04	=	**1.06E+04**	**1.19E+04**	-	1.49E+04	1.53E+04	=
20	1.31E+03	3.25E+02	1.40E+03	3.69E+02	=	1.71E+03	3.98E+02	+	1.07E+03	2.83E+02	-	**8.63E+02**	**2.40E+02**	-	2.56E+03	6.28E+02	+	1.64E+03	4.06E+02	+	1.08E+03	3.20E+02	-
21	4.23E+02	4.67E+01	4.62E+02	4.92E+01	+	6.96E+02	5.81E+01	+	5.26E+02	4.87E+01	+	**3.90E+02**	**3.19E+01**	-	1.27E+03	1.83E+02	+	6.08E+02	7.82E+01	+	5.02E+02	6.82E+01	+
22	**5.51E+03**	**6.28E+02**	5.98E+03	6.73E+02	+	1.08E+04	1.25E+03	+	8.17E+03	8.39E+02	+	8.00E+03	1.36E+03	+	9.40E+03	1.08E+03	+	7.30E+03	7.86E+02	+	6.35E+03	6.72E+02	+
23	**6.86E+02**	**4.67E+01**	7.13E+02	5.01E+01	+	9.16E+02	6.39E+01	+	7.79E+02	4.78E+01	+	7.29E+02	5.42E+01	+	2.97E+03	8.65E+02	+	8.73E+02	8.13E+01	+	7.87E+02	6.27E+01	+
24	1.18E+03	1.56E+02	1.26E+03	1.86E+02	=	9.18E+02	7.15E+01	-	**8.20E+02**	**4.60E+01**	-	8.37E+02	7.78E+01	-	1.72E+03	2.16E+02	+	1.33E+03	1.66E+02	+	1.14E+03	1.20E+02	=
25	**5.28E+02**	**6.24E+01**	5.31E+02	3.70E+01	=	7.53E+03	4.80E+03	+	1.31E+03	3.28E+02	+	6.07E+02	3.39E+01	+	5.56E+02	3.20E+01	+	5.44E+02	4.02E+01	=	5.30E+02	4.10E+01	=
26	**3.73E+03**	**6.90E+02**	3.85E+03	7.26E+02	=	6.27E+03	9.78E+02	+	4.79E+03	5.59E+02	+	4.12E+03	4.47E+02	+	1.50E+04	3.87E+03	+	5.58E+03	1.13E+03	+	4.87E+03	7.95E+02	+
27	7.91E+02	1.09E+02	8.48E+02	1.34E+02	+	9.89E+02	1.35E+02	+	**7.54E+02**	**8.14E+01**	=	1.09E+03	8.94E+01	+	2.84E+03	1.39E+03	+	1.02E+03	1.17E+02	+	8.81E+02	1.29E+02	+
28	4.93E+02	2.54E+01	**4.86E+02**	**2.30E+01**	=	4.52E+03	1.29E+03	+	1.46E+03	6.39E+02	+	6.65E+02	4.54E+01	+	5.04E+02	1.25E+01	+	5.03E+02	1.58E+01	+	4.96E+02	1.86E+01	+
29	**1.48E+03**	**2.63E+02**	1.60E+03	3.87E+02	=	3.11E+03	8.07E+02	+	1.50E+03	3.47E+02	=	1.70E+03	4.36E+02	+	2.91E+03	4.35E+02	+	1.87E+03	4.13E+02	+	1.64E+03	3.08E+02	+
30	1.24E+06	4.72E+05	1.13E+06	3.50E+05	=	2.30E+08	2.48E+08	+	2.26E+06	8.55E+05	+	3.87E+07	1.73E+06	+	9.73E+05	3.49E+05	-	9.83E+05	2.18E+05	-	**9.71E+05**	**3.17E+05**	-

Table 11 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against SNUM, rcGA, cDE, nuSA, ISPO, ISPOR and VISPO on CEC-2017 [2] in 100 dimensions.

Function	cSNUM		SNUM			rcGA			cDE			nuSA			ISPO			ISPOR			VISPO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	2.24E+04	2.33E+04	2.70E+04	2.59E+04	=	2.16E+11	4.04E+10	+	7.89E+10	1.75E+10	+	6.62E+09	5.31E+08	+	1.89E+04	1.94E+04	=	**1.38E+04**	**1.77E+04**	-	1.24E+05	7.68E+05	=
2	1.29E+93	9.21E+93	9.32E+108	6.66E+109	+	2.01E+163	Inf	+	2.23E+146	1.45E+147	+	2.01E+97	8.94E+97	+	**4.66E+37**	**3.33E+38**	-	1.08E+38	7.67E+38	-	7.65E+102	5.46E+103	+
3	1.85E+05	5.08E+04	5.99E+05	8.45E+04	+	8.55E+05	1.36E+05	+	6.01E+05	8.31E+04	+	**8.32E+04**	**4.18E+03**	-	6.76E+05	1.24E+05	+	6.61E+05	1.19E+05	+	4.39E+05	3.20E+05	=
4	2.05E+02	3.85E+01	2.29E+02	3.95E+01	+	3.81E+04	1.64E+04	+	1.26E+04	4.14E+03	+	9.96E+02	4.90E+01	+	2.09E+02	3.43E+01	=	**2.02E+02**	**3.61E+01**	=	2.16E+02	2.94E+01	=
5	6.16E+02	1.03E+02	6.53E+02	1.18E+02	=	1.47E+03	1.74E+02	+	1.16E+03	1.06E+02	+	**4.82E+02**	**3.96E+01**	-	2.14E+03	2.70E+02	+	1.06E+03	1.58E+02	+	6.23E+02	1.35E+02	=
6	3.23E-06	1.18E-06	**2.57E-07**	**1.82E-06**	-	9.40E+01	1.11E+01	+	5.63E+01	5.89E+00	+	4.36E+01	4.40E+00	+	9.40E+01	5.77E+00	+	1.95E+00	4.31E-01	+	3.60E-04	2.57E-03	-
7	**5.97E+02**	**5.75E+01**	6.94E+02	8.99E+01	+	8.98E+03	6.78E+02	+	4.80E+03	3.69E+02	+	7.91E+02	1.01E+02	+	1.36E+04	1.48E+03	+	1.48E+03	1.88E+02	+	1.06E+03	1.31E+02	+
8	6.25E+02	1.09E+02	6.95E+02	8.52E+01	+	1.48E+03	1.62E+02	+	1.13E+03	1.04E+02	+	**5.16E+02**	**5.47E+01**	-	2.30E+03	3.01E+02	+	1.14E+03	1.67E+02	+	6.97E+02	1.43E+02	+
9	2.58E+04	6.80E+03	2.93E+04	7.17E+03	+	8.40E+04	1.76E+04	+	5.45E+04	1.23E+04	+	**1.61E+04**	**5.90E+03**	-	5.77E+04	5.77E+03	+	4.88E+04	6.87E+03	+	3.40E+04	1.09E+04	+
10	**1.22E+04**	**1.11E+03**	1.27E+04	1.17E+03	+	2.51E+04	1.75E+03	+	2.14E+04	1.33E+03	+	1.76E+04	2.83E+03	+	1.78E+04	1.64E+03	+	1.50E+04	1.34E+03	+	1.38E+04	1.18E+03	+
11	**9.02E+02**	**8.25E+02**	3.70E+04	1.80E+04	+	2.53E+05	7.33E+04	+	8.01E+04	2.35E+04	+	3.36E+03	2.35E+02	+	8.16E+04	4.04E+04	+	9.03E+04	4.50E+04	+	9.07E+04	3.12E+04	+
12	**2.37E+06**	**8.59E+05**	4.97E+06	1.94E+06	+	4.85E+10	1.84E+10	+	1.72E+10	5.29E+09	+	3.64E+08	3.44E+07	+	7.21E+06	4.05E+06	+	6.88E+06	4.61E+06	+	8.27E+06	3.41E+06	+
13	9.62E+03	9.90E+03	**8.92E+03**	**9.19E+03**	=	4.62E+09	2.94E+09	+	2.76E+08	3.08E+08	+	2.94E+04	5.12E+03	+	2.18E+04	1.34E+04	+	1.09E+04	9.79E+03	=	9.68E+03	8.92E+03	=
14	5.08E+05	2.73E+05	2.01E+06	9.42E+05	+	3.89E+07	2.49E+07	+	3.63E+06	2.57E+06	+	**1.89E+05**	**6.60E+04**	-	1.92E+06	9.55E+05	+	1.93E+06	9.33E+05	+	1.02E+07	5.03E+06	+
15	5.23E+03	6.85E+03	9.37E+03	8.81E+03	+	1.26E+09	1.28E+09	+	1.67E+05	5.78E+05	+	3.04E+04	8.12E+03	+	5.80E+03	5.00E+03	=	**3.83E+03**	**4.20E+03**	=	1.68E+04	4.67E+04	=
16	4.48E+03	6.73E+02	4.82E+03	8.69E+02	+	8.21E+03	1.18E+03	+	6.61E+03	9.08E+02	+	**4.19E+03**	**7.97E+02**	=	5.45E+03	7.83E+02	+	5.36E+03	6.89E+02	+	4.80E+03	8.76E+02	=
17	3.60E+03	5.52E+02	3.71E+03	5.98E+02	=	1.10E+04	8.10E+03	+	4.44E+03	6.25E+02	+	**2.64E+03**	**4.65E+02**	-	5.15E+03	8.00E+02	+	4.56E+03	6.90E+02	+	3.89E+03	6.34E+02	+
18	6.76E+05	3.16E+05	2.34E+06	1.42E+06	+	4.96E+07	4.86E+07	+	1.05E+07	8.22E+06	+	**2.19E+05**	**5.62E+04**	-	1.80E+06	7.28E+05	+	1.94E+06	9.40E+05	+	5.97E+06	4.84E+06	+
19	8.39E+03	9.70E+03	1.01E+04	8.80E+03	=	1.61E+09	1.25E+09	+	2.00E+07	2.60E+07	+	6.32E+05	1.53E+05	+	5.96E+03	6.28E+03	=	**5.54E+03**	**5.45E+03**	=	7.07E+03	8.74E+03	=
20	3.29E+03	6.08E+02	3.50E+03	5.78E+02	=	4.66E+03	6.77E+02	+	3.58E+03	5.48E+02	+	**2.67E+03**	**5.14E+02**	-	5.24E+03	7.81E+02	+	4.19E+03	5.74E+02	+	3.41E+03	6.55E+02	=
21	**8.80E+02**	**9.19E+01**	9.34E+02	7.78E+01	+	1.70E+03	1.24E+02	+	1.39E+03	7.89E+01	+	8.84E+02	7.67E+01	=	2.72E+03	3.42E+02	+	1.31E+03	1.33E+02	+	1.03E+03	1.25E+02	+
22	**1.30E+04**	**1.27E+03**	1.43E+04	1.18E+03	+	2.60E+04	1.84E+03	+	2.23E+04	1.34E+03	+	1.98E+04	2.84E+03	+	2.00E+04	1.91E+03	+	1.67E+04	1.50E+03	+	1.52E+04	1.34E+03	+
23	**8.95E+02**	**5.88E+01**	9.22E+02	6.40E+01	+	1.73E+03	1.29E+02	+	1.43E+03	7.33E+01	+	1.68E+03	1.04E+02	+	4.34E+03	1.13E+03	+	1.12E+03	1.12E+02	+	1.03E+03	7.50E+01	+
24	**1.47E+03**	**8.18E+01**	1.52E+03	9.43E+01	+	2.25E+03	1.91E+02	+	1.93E+03	8.85E+01	+	2.49E+03	1.38E+02	+	4.26E+03	4.04E+02	+	1.80E+03	1.34E+02	+	1.68E+03	1.13E+02	+
25	**7.47E+02**	**5.74E+01**	7.72E+02	5.46E+01	+	3.72E+04	1.05E+04	+	1.25E+04	2.83E+03	+	1.21E+03	3.67E+01	+	7.88E+02	7.05E+01	+	7.83E+02	5.85E+01	+	7.63E+02	5.32E+01	+
26	**9.87E+03**	**1.01E+03**	1.04E+04	9.61E+02	+	1.85E+04	2.47E+03	+	1.49E+04	9.64E+02	+	1.47E+04	7.85E+02	+	3.49E+04	3.19E+03	+	1.44E+04	1.53E+03	+	1.21E+04	1.25E+03	+
27	8.71E+02	8.16E+01	**8.35E+02**	**6.64E+01**	-	1.70E+03	3.01E+02	+	1.23E+03	1.32E+02	+	1.89E+03	1.31E+02	+	2.72E+03	2.07E+03	+	1.07E+03	1.39E+02	+	9.32E+02	1.00E+02	+
28	5.50E+02	3.23E+01	5.56E+02	3.68E+01	=	2.10E+04	4.09E+03	+	1.21E+04	2.93E+03	+	1.61E+03	7.44E+01	+	**5.37E+02**	**4.32E+01**	=	7.91E+02	1.71E+03	=	5.70E+02	3.87E+01	+
29	**3.78E+03**	**5.46E+02**	5.00E+03	6.79E+02	+	1.78E+04	1.11E+04	+	5.63E+03	7.44E+02	+	5.07E+03	4.59E+02	+	6.64E+03	8.99E+02	+	5.65E+03	7.38E+02	+	5.00E+03	5.19E+02	+
30	**9.01E+03**	**5.37E+03**	1.42E+04	5.88E+03	+	2.25E+09	2.13E+09	+	2.20E+08	1.87E+08	+	2.76E+07	4.03E+06	+	1.73E+04	4.92E+03	+	1.70E+04	4.84E+03	+	1.65E+04	4.77E+03	+

C Extended numerical results for cSNUM on CEC-2017 against population-based algorithms

Table 12 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL and SNUM on CEC-2017 [2] in 10 dimensions.

Function	cSNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL			SNUM		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	4.93E+03	4.00E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.54E-09	7.33E-09	-	2.41E+02	2.02E+02	-	2.10E+03	2.50E+03	-	5.58E+03	4.36E+03	=
2	4.16E-04	7.20E-04	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	2.16E-01	1.54E+00	-	3.92E-02	1.96E-01	-	1.61E+02	5.94E+02	-
3	8.19E-13	1.45E-12	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.14E-04	4.29E-04	+	1.94E+01	1.10E+01	+	4.03E+02	5.87E+02	+
4	3.27E+00	1.38E+01	1.24E-03	3.07E-03	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.90E+00	1.46E+00	+	3.97E+00	1.21E+00	+	1.32E+01	2.54E+01	+
5	1.37E+01	5.07E+00	**1.83E+00**	**1.60E+00**	-	9.31E+00	2.39E+00	-	5.39E+00	1.18E+00	-	2.13E+00	2.03E+00	-	1.81E+01	5.10E+00	+	1.63E+01	5.29E+00	+	2.17E+01	8.56E+00	+
6	7.49E-07	2.05E-06	7.58E-08	3.64E-08	-	3.57E-05	1.53E-05	+	**0.00E+00**	**0.00E+00**	-	1.34E-04	7.12E-04	-	9.18E-01	1.87E+00	+	8.40E-08	4.44E-07	-	6.57E-05	2.34E-04	-
7	1.81E+01	3.76E+00	1.16E+01	6.16E-01	-	2.27E+01	2.89E+00	+	1.43E+01	1.29E+00	-	**1.11E+01**	**2.61E+00**	-	1.78E+01	2.36E+00	=	3.22E+01	3.79E+00	+	3.20E+01	7.82E+00	+
8	1.29E+01	5.85E+00	**1.68E+00**	**1.16E+00**	-	1.04E+01	2.56E+00	=	4.26E+00	1.34E+00	-	3.22E+00	3.29E+00	-	9.95E+00	2.37E+00	-	2.01E+01	4.38E+00	+	2.50E+01	1.06E+01	+
9	3.01E+00	2.15E+01	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	2.80E-04	1.49E-03	-	8.91E-03	6.36E-02	-	3.72E+01	1.58E+02	+
10	4.52E+02	2.14E+02	3.19E+02	1.19E+02	-	7.01E+02	1.78E+02	+	**2.86E+02**	**1.01E+02**	-	5.39E+02	2.12E+02	+	5.03E+02	1.55E+02	=	1.14E+03	2.07E+02	+	6.86E+02	2.65E+02	+
11	7.12E+00	4.52E+00	1.67E+00	1.02E+00	-	3.22E+00	5.52E-01	-	**1.19E-06**	**8.51E-06**	-	9.03E-01	8.58E-01	-	1.73E+01	5.53E+00	+	4.49E+00	1.49E+00	-	1.86E+01	2.67E+01	+
12	1.97E+04	1.56E+04	9.82E+01	7.08E+01	-	**2.44E+01**	**1.96E+01**	-	1.19E+02	7.16E+01	-	2.19E+02	1.50E+02	-	4.76E+03	2.58E+03	-	1.38E+05	1.60E+05	+	1.51E+05	2.06E+05	+
13	7.51E+03	9.02E+03	**4.62E+00**	**2.97E+00**	-	7.74E+00	1.88E+00	-	4.94E+00	2.53E+00	-	8.80E+00	2.65E+01	-	1.46E+03	1.41E+03	-	3.20E+03	2.76E+03	=	7.96E+03	1.08E+04	=
14	8.17E+03	8.55E+03	3.67E+00	3.46E+00	-	7.18E+00	1.89E+00	-	**3.32E+00**	**7.28E+00**	-	2.52E+01	1.23E+01	-	6.29E+01	6.84E+01	-	9.46E+01	3.37E+01	-	1.03E+04	9.57E+03	=
15	6.11E+03	8.54E+03	**2.69E-01**	**1.27E-01**	-	6.38E-01	1.94E-01	-	4.08E-01	2.29E-01	-	3.14E+01	4.19E+01	-	1.87E+02	1.86E+02	-	2.14E+02	7.54E+01	-	1.18E+04	1.16E+04	+
16	1.70E+02	1.39E+02	3.42E+00	6.35E-01	-	5.41E+00	1.44E+00	-	**1.41E+00**	**8.39E-01**	-	1.49E+02	1.32E+02	=	8.36E+01	7.26E+01	-	1.23E+01	2.40E+01	-	2.17E+02	1.63E+02	=
17	5.13E+01	6.38E+01	1.21E+01	3.80E+00	-	3.44E+01	3.72E+00	=	**1.15E+01**	**6.34E+00**	=	6.61E+01	3.86E+01	+	2.47E+01	7.47E+00	=	4.65E+01	9.62E+00	+	5.84E+01	7.14E+01	=
18	1.30E+04	1.22E+04	1.41E+00	4.24E+00	-	**4.84E-01**	**2.04E-01**	-	5.96E+00	9.00E+00	-	4.43E+01	5.10E+01	-	2.26E+03	2.19E+03	-	9.41E+03	6.52E+03	=	2.08E+04	1.40E+04	+
19	1.04E+04	1.13E+04	**2.23E-01**	**1.37E-01**	-	1.19E+00	3.34E-01	-	2.35E-01	3.79E-01	-	3.63E+00	3.65E+00	-	1.48E+02	2.38E+02	-	1.24E+02	1.08E+02	-	1.00E+04	1.09E+04	=
20	4.27E+00	6.90E+00	**1.74E+00**	**2.03E+00**	=	2.15E+01	3.97E+00	+	2.71E+00	6.51E+00	-	1.23E+02	7.83E+01	+	2.98E+01	1.09E+01	+	2.26E+01	9.74E+00	+	1.07E+01	9.14E+00	+
21	2.16E+02	3.03E+01	1.39E+02	3.97E+01	-	1.40E+02	5.13E+01	-	**1.02E+02**	**1.01E+01**	-	1.55E+02	5.24E+01	-	1.05E+02	2.15E+01	-	1.48E+02	5.46E+01	-	2.21E+02	4.63E+01	+
22	2.28E+02	2.96E+02	9.85E+01	1.10E+01	-	1.00E+02	1.80E-01	-	1.00E+02	1.13E-01	-	9.93E+01	5.57E+00	-	9.68E+01	1.68E+01	-	**9.37E+01**	**2.22E+01**	-	7.52E+02	5.36E+02	+
23	3.19E+02	7.67E+00	3.04E+02	2.01E+00	-	3.07E+02	2.15E+00	-	3.06E+02	1.24E+00	-	**2.87E+02**	**5.52E+01**	-	3.42E+02	1.05E+01	+	3.11E+02	6.41E+00	-	3.31E+02	2.44E+01	+
24	3.60E+02	5.23E+01	**2.14E+02**	**8.48E+01**	-	3.28E+02	3.57E+01	-	2.90E+02	8.55E+01	-	2.33E+02	1.10E+02	-	2.27E+02	1.35E+02	-	3.14E+02	6.72E+01	-	3.40E+02	8.58E+01	=
25	4.32E+02	3.14E+01	4.13E+02	2.10E+01	-	4.06E+02	1.75E+01	-	4.28E+02	2.16E+01	-	4.07E+02	6.65E+01	-	**4.04E+02**	**1.45E+01**	-	4.26E+02	2.25E+01	-	4.21E+02	6.10E+01	=
26	8.47E+02	5.45E+02	**2.66E+02**	**4.67E+01**	-	3.00E+02	0.00E+00	-	3.00E+02	0.00E+00	-	2.85E+02	1.45E+02	-	2.67E+02	7.66E+01	-	3.02E+02	4.66E+01	-	1.02E+03	5.72E+02	+
27	4.06E+02	2.33E+01	3.92E+02	2.40E+00	-	**3.89E+02**	**2.22E-01**	-	3.90E+02	9.32E-01	-	3.95E+02	3.98E+00	-	4.27E+02	1.35E+01	+	3.93E+02	3.35E+00	-	4.37E+02	3.53E+01	+
28	5.44E+02	1.44E+02	3.22E+02	9.83E+01	-	3.43E+02	1.00E+02	-	3.17E+02	6.97E+01	-	4.18E+02	1.71E+02	-	**2.94E+02**	**4.21E+01**	-	4.49E+02	1.56E+02	-	5.67E+02	1.29E+02	=
29	3.20E+02	6.19E+01	**2.45E+02**	**5.82E+00**	-	2.57E+02	6.87E+00	-	2.48E+02	6.52E+00	-	2.87E+02	5.30E+01	-	2.78E+02	1.36E+01	-	2.87E+02	1.63E+01	=	3.76E+02	8.02E+01	+
30	2.56E+05	4.19E+05	4.09E+02	1.82E+01	-	**3.95E+02**	**1.01E-01**	-	4.30E+02	2.59E+01	-	2.29E+03	1.04E+04	-	7.09E+03	3.12E+03	-	2.98E+05	5.08E+05	=	5.02E+05	6.81E+05	+

Table 13 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL and SNUM on CEC-2017 [2] in 30 dimensions.

Function	cSNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL			SNUM		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	7.00E+03	6.93E+03	1.25E-04	8.32E-05	-	3.68E-06	1.95E-06	-	**0.00E+00**	**0.00E+00**	-	6.27E-08	1.93E-08	-	7.55E+02	6.11E+02	-	3.66E+03	3.76E+03	-	9.53E+03	7.62E+03	+
2	1.41E-03	2.88E-03	4.45E+00	1.40E+01	-	**1.72E-08**	**6.48E-08**	-	1.96E-02	1.40E-01	-	1.17E+02	8.35E+02	-	9.50E+06	1.71E+07	+	8.52E+16	5.81E+17	+	3.69E+23	2.41E+24	+
3	7.02E+03	7.77E+03	3.59E+01	2.40E+02	-	4.03E-08	1.98E-08	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	4.08E+02	2.03E+02	-	2.08E+04	4.64E+03	+	1.04E+05	3.83E+04	+
4	6.99E+01	2.55E+01	6.93E+01	1.17E+01	=	5.87E+01	7.78E-01	-	5.91E+01	3.45E+00	-	**5.53E+01**	**1.65E+01**	-	5.88E+01	3.40E+01	-	9.49E+01	2.35E+01	+	7.37E+01	2.73E+01	+
5	9.09E+01	2.04E+01	1.17E+01	3.96E+00	-	1.25E+02	1.07E+01	+	3.03E+01	4.32E+01	-	**4.00E+00**	**1.94E+00**	-	1.12E+02	1.33E+01	+	7.57E+01	4.35E+01	-	1.16E+02	3.45E+01	+
6	9.69E-07	4.90E-07	4.60E-05	1.42E-05	+	2.92E-04	5.66E-05	+	**1.31E-07**	**9.11E-08**	-	5.40E-05	3.78E-04	-	2.16E+01	3.97E+00	+	4.87E-01	4.24E-01	+	6.68E-05	3.34E-04	-
7	1.05E+02	1.96E+01	3.94E+01	1.23E+01	-	1.70E+02	1.33E+01	+	4.50E+01	2.47E+01	-	**3.52E+01**	**1.81E+00**	-	1.35E+02	1.62E+01	+	1.89E+02	2.09E+01	+	1.36E+02	2.41E+01	+
8	1.11E+02	2.72E+01	2.39E+01	3.50E+01	-	1.25E+02	1.23E+01	+	6.99E+00	1.93E+01	-	**4.35E+00**	**2.66E+00**	-	8.10E+01	1.04E+01	-	7.14E+01	4.37E+01	-	1.34E+02	3.45E+01	+
9	1.77E+03	1.78E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.38E+03	2.72E+02	=	3.46E+01	2.73E+01	-	2.62E+03	1.93E+03	+
10	2.69E+03	4.36E+02	4.40E+03	7.91E+02	+	6.19E+03	3.00E+02	+	5.28E+03	8.28E+02	+	**1.87E+03**	**7.56E+02**	-	3.13E+03	3.46E+02	+	6.99E+03	2.62E+02	+	3.22E+03	5.03E+02	+
11	7.59E+01	3.64E+01	4.40E+01	1.78E+01	-	3.19E+01	1.35E+01	-	**1.78E+01**	**2.48E+01**	-	6.32E+01	3.96E+01	=	8.83E+01	2.12E+01	+	8.99E+01	4.19E+01	=	1.15E+02	4.81E+01	+
12	5.75E+05	4.36E+05	2.53E+03	1.27E+03	-	**5.78E+02**	**3.04E+02**	-	6.15E+02	3.13E+02	-	1.38E+03	3.62E+02	-	9.51E+04	5.13E+04	-	1.04E+05	1.70E+05	-	1.31E+06	1.09E+06	+
13	1.95E+04	2.01E+04	8.02E+01	1.96E+01	-	6.68E+01	8.38E+00	-	**2.90E+01**	**2.42E+01**	-	9.97E+02	9.09E+02	-	3.21E+03	2.88E+03	-	2.06E+04	1.78E+04	=	2.45E+04	2.55E+04	=
14	1.18E+05	1.37E+05	6.36E+01	3.95E+00	-	4.86E+01	5.59E+00	-	**2.61E+01**	**8.63E+00**	-	1.74E+02	4.94E+01	-	3.48E+03	2.63E+03	-	1.13E+04	8.70E+03	-	9.20E+05	1.07E+06	+
15	1.62E+04	1.49E+04	3.75E+01	5.32E+00	-	2.36E+01	3.04E+00	-	**6.51E+00**	**2.41E+00**	-	2.75E+02	1.57E+02	-	2.13E+03	1.63E+03	-	2.90E+04	2.57E+04	+	2.12E+04	1.46E+04	+
16	1.27E+03	3.34E+02	9.04E+02	2.05E+02	-	8.20E+02	1.79E+02	-	**4.75E+02**	**3.40E+02**	-	5.83E+02	2.48E+02	-	8.46E+02	1.53E+02	-	7.48E+02	4.33E+02	-	1.24E+03	3.53E+02	=
17	7.14E+02	2.54E+02	1.78E+02	3.36E+01	-	2.14E+02	2.57E+01	-	1.88E+02	4.84E+01	-	2.48E+02	1.60E+02	-	3.31E+02	1.13E+02	-	**1.52E+02**	**7.04E+01**	-	7.61E+02	2.49E+02	=
18	5.49E+05	4.58E+05	4.19E+01	4.31E+00	-	3.05E+01	1.84E+00	-	**2.39E+01**	**1.92E+00**	-	2.20E+02	1.23E+02	-	8.77E+04	3.31E+04	-	5.31E+05	2.20E+05	=	3.14E+06	3.50E+06	+
19	1.36E+04	1.22E+04	2.26E+01	2.14E+00	-	2.41E+01	1.70E+00	-	**1.08E+01**	**2.94E+00**	-	1.91E+02	7.37E+01	-	1.71E+03	1.69E+03	-	1.23E+04	1.13E+04	=	1.67E+04	1.74E+04	=
20	5.84E+02	2.33E+02	2.64E+02	5.36E+01	-	3.12E+02	4.74E+01	-	**1.56E+02**	**1.01E+02**	-	4.97E+02	2.03E+02	=	3.58E+02	1.17E+02	-	3.33E+02	1.37E+02	-	7.31E+02	2.96E+02	+
21	3.18E+02	3.44E+01	2.16E+02	1.59E+01	-	3.19E+02	9.95E+00	=	2.83E+02	5.58E+01	-	**2.13E+02**	**4.75E+00**	-	3.05E+02	3.30E+01	=	2.65E+02	4.14E+01	-	3.39E+02	3.55E+01	+
22	2.59E+03	1.25E+03	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	1.09E+03	1.12E+03	-	**1.00E+02**	**2.36E-03**	-	1.01E+02	1.94E+00	-	3.49E+03	1.04E+03	+
23	4.61E+02	2.72E+01	3.80E+02	2.38E+01	-	4.67E+02	1.08E+01	+	4.55E+02	1.47E+01	=	**3.47E+02**	**5.02E+01**	-	6.81E+02	3.79E+01	+	3.97E+02	1.62E+01	-	4.78E+02	2.91E+01	+
24	7.04E+02	1.04E+02	4.54E+02	3.48E+01	-	5.31E+02	8.43E+00	-	5.41E+02	2.90E+01	-	**4.23E+02**	**3.24E+00**	-	7.39E+02	4.57E+01	+	4.72E+02	2.00E+01	-	7.03E+02	1.11E+02	=
25	3.93E+02	1.61E+01	3.87E+02	2.14E-02	-	3.87E+02	7.72E-03	-	3.87E+02	1.06E-02	-	3.87E+02	1.49E-02	-	**3.86E+02**	**1.77E+00**	-	4.02E+02	1.76E+01	+	3.98E+02	2.12E+01	+
26	2.28E+03	6.72E+02	7.75E+02	3.46E+02	-	2.00E+03	9.68E+01	-	1.09E+03	3.17E+02	-	**5.09E+02**	**2.71E+02**	-	2.04E+03	1.73E+03	=	1.43E+03	4.62E+02	-	2.37E+03	6.26E+02	=
27	5.35E+02	1.98E+01	5.02E+02	4.03E+00	-	**4.99E+02**	**7.99E+00**	-	5.05E+02	6.37E+00	-	5.20E+02	1.32E+01	-	7.08E+02	5.41E+01	+	5.32E+02	2.07E+01	=	5.42E+02	1.75E+01	+
28	3.79E+02	6.74E+01	3.61E+02	4.43E+01	=	3.13E+02	3.36E+01	-	3.14E+02	3.52E+01	-	**3.13E+02**	**3.48E+01**	=	3.91E+02	1.77E+01	=	4.35E+02	2.69E+01	+	4.31E+02	4.04E+01	+
29	9.79E+02	2.32E+02	7.33E+02	8.42E+01	-	7.51E+02	3.73E+01	-	**4.89E+02**	**7.49E+01**	-	5.68E+02	1.54E+02	-	7.81E+02	1.21E+02	-	6.26E+02	9.19E+01	-	1.11E+03	2.58E+02	+
30	1.11E+04	1.84E+04	2.04E+03	6.20E+01	-	**1.98E+03**	**1.88E+01**	-	2.01E+03	6.24E+01	-	3.07E+03	1.93E+03	-	5.69E+03	1.94E+03	-	3.65E+04	3.88E+04	+	1.60E+04	1.13E+04	+

Table 14 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL and SNUM on CEC-2017 [2] in 50 dimensions.

Function	cSNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL			SNUM		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.42E+04	1.30E+04	7.30E+02	4.56E+02	-	2.44E-01	2.97E-01	-	2.57E-02	2.86E-02	-	**1.23E-07**	**4.41E-08**	-	4.97E+03	2.50E+03	-	1.02E+06	3.03E+06	+	1.34E+04	1.25E+04	=
2	1.49E+20	1.06E+21	1.25E+04	4.86E+04	-	3.77E+04	1.96E+05	=	**1.87E+02**	**5.84E+02**	-	2.77E+05	1.98E+06	-	1.25E+18	4.36E+18	+	1.57E+39	1.01E+40	+	6.62E+35	4.72E+36	+
3	2.09E+04	1.27E+04	3.51E+03	1.04E+04	-	2.18E-04	1.56E-04	-	3.57E-08	3.23E-08	-	**0.00E+00**	**0.00E+00**	-	1.54E+04	3.23E+03	=	7.02E+04	9.11E+03	+	2.24E+05	5.44E+04	+
4	8.46E+01	5.06E+01	6.83E+01	4.60E+01	=	8.19E+01	5.67E+01	=	7.53E+01	5.51E+01	=	**3.82E+01**	**4.42E+01**	-	1.11E+02	4.11E+01	+	1.98E+02	4.63E+01	+	1.14E+02	5.90E+01	+
5	2.12E+02	3.76E+01	1.11E+02	1.04E+02	-	2.72E+02	1.16E+01	+	**5.99E+00**	**2.02E+00**	-	7.47E+00	2.74E+00	-	2.01E+02	1.37E+01	=	1.01E+02	1.80E+01	-	2.29E+02	4.90E+01	=
6	1.52E-06	6.20E-07	2.24E-04	2.35E-05	+	9.82E-05	2.94E-05	+	1.46E-05	6.24E-06	+	**1.89E-07**	**1.72E-07**	-	3.37E+01	3.92E+00	+	4.52E+00	1.70E+00	+	4.23E-05	2.12E-04	-
7	2.04E+02	2.93E+01	1.22E+02	5.00E+01	-	3.39E+02	1.56E+01	+	6.08E+01	2.56E+01	-	**5.93E+01**	**2.09E+00**	-	2.78E+02	3.42E+01	+	2.60E+02	1.04E+02	=	2.58E+02	3.91E+01	+
8	2.11E+02	4.96E+01	1.46E+02	1.18E+02	-	2.71E+02	1.76E+01	+	**5.81E+00**	**1.88E+00**	-	7.94E+00	2.54E+00	-	1.99E+02	1.52E+01	=	9.73E+01	1.54E+01	-	2.40E+02	5.43E+01	+
9	6.09E+03	4.05E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	6.13E+03	7.42E+02	=	1.73E+03	1.19E+03	-	6.60E+03	3.91E+03	=
10	4.85E+03	8.05E+02	7.70E+03	9.00E+02	+	1.21E+04	3.37E+02	+	9.37E+03	3.57E+03	+	**3.00E+03**	**9.51E+02**	-	5.20E+03	5.52E+02	+	1.29E+04	4.00E+02	+	5.25E+03	7.71E+02	+
11	1.52E+02	4.33E+01	1.32E+02	9.01E+00	-	9.67E+01	8.37E+00	-	**3.24E+01**	**4.63E+00**	-	2.13E+02	5.70E+01	+	1.29E+02	1.43E+01	-	1.74E+02	4.91E+01	+	1.62E+03	2.43E+03	+
12	1.84E+06	1.34E+06	2.97E+04	1.34E+04	-	1.75E+03	5.18E+02	-	**1.60E+03**	**4.19E+02**	-	3.89E+06	2.76E+07	-	1.01E+06	3.45E+05	-	1.30E+06	1.05E+06	=	2.62E+06	1.48E+06	+
13	1.01E+04	1.02E+04	3.04E+02	3.78E+01	-	1.94E+02	3.40E+01	-	**1.57E+02**	**4.53E+01**	-	2.32E+03	1.60E+03	-	9.01E+02	5.62E+02	-	8.40E+03	5.19E+03	=	1.65E+04	1.42E+04	+
14	1.92E+05	1.05E+05	1.36E+02	9.33E+00	-	1.01E+02	7.72E+00	-	**3.11E+01**	**4.86E+00**	-	2.93E+02	8.35E+01	-	2.60E+04	1.34E+04	-	1.25E+05	6.54E+04	-	7.60E+05	5.23E+05	+
15	1.16E+04	9.03E+03	1.27E+02	1.27E+01	-	8.36E+01	8.41E+00	-	**3.08E+01**	**5.59E+00**	-	8.25E+02	2.08E+02	-	1.20E+03	7.81E+02	-	6.97E+03	6.03E+03	-	1.48E+04	8.95E+03	=
16	2.13E+03	3.66E+02	1.93E+03	3.90E+02	-	1.96E+03	1.98E+02	-	1.02E+03	6.77E+02	-	**9.44E+02**	**3.40E+02**	-	1.25E+03	2.31E+02	-	9.78E+02	3.64E+02	-	2.27E+03	4.95E+02	=
17	1.62E+03	3.14E+02	1.47E+03	2.16E+02	-	1.32E+03	1.29E+02	-	**6.34E+02**	**4.50E+02**	-	8.46E+02	2.34E+02	-	1.03E+03	1.54E+02	-	1.16E+03	4.90E+02	-	1.59E+03	3.82E+02	=
18	6.88E+05	4.33E+05	1.17E+02	1.73E+01	-	5.39E+01	5.61E+00	-	**3.31E+01**	**6.10E+00**	-	3.87E+02	1.45E+02	-	3.36E+05	1.46E+05	-	1.74E+06	9.40E+05	+	2.78E+06	2.04E+06	+
19	1.68E+04	1.53E+04	7.10E+01	5.98E+00	-	5.07E+01	4.64E+00	-	**2.42E+01**	**6.98E+00**	-	2.03E+02	5.75E+01	-	8.67E+03	3.91E+03	=	1.45E+04	9.40E+03	=	1.94E+04	1.70E+04	=
20	1.31E+03	3.25E+02	1.34E+03	2.35E+02	=	1.16E+03	1.22E+02	-	**3.32E+02**	**3.33E+02**	-	9.05E+02	2.79E+02	-	7.80E+02	1.97E+02	-	1.31E+03	2.81E+02	=	1.40E+03	3.69E+02	=
21	4.23E+02	4.67E+01	3.17E+02	1.16E+02	-	4.73E+02	1.16E+01	+	4.75E+02	5.98E+01	+	**2.12E+02**	**3.57E+00**	-	4.33E+02	2.14E+01	=	2.93E+02	3.32E+01	-	4.62E+02	4.92E+01	+
22	5.51E+03	6.28E+02	**2.14E+03**	**3.56E+03**	-	7.55E+03	5.81E+03	+	2.44E+03	4.09E+03	-	3.48E+03	1.92E+03	-	5.98E+03	9.89E+02	+	6.99E+03	6.55E+03	=	5.98E+03	6.73E+02	+
23	6.86E+02	4.67E+01	6.84E+02	1.02E+02	+	6.89E+02	1.78E+01	=	7.16E+02	2.78E+01	+	**4.33E+02**	**1.41E+01**	-	1.07E+03	7.19E+01	+	5.67E+02	3.41E+01	-	7.13E+02	5.01E+01	+
24	1.18E+03	1.56E+02	7.42E+02	1.22E+02	-	7.43E+02	1.87E+01	-	8.09E+02	2.32E+01	-	**4.95E+02**	**6.45E+00**	-	1.08E+03	7.07E+01	-	6.78E+02	4.67E+01	-	1.26E+03	1.86E+02	=
25	5.28E+02	6.24E+01	4.90E+02	2.67E+01	-	4.81E+02	2.80E+00	-	**4.81E+02**	**2.32E+00**	-	4.85E+02	1.71E+01	-	5.55E+02	2.65E+01	+	6.21E+02	2.61E+01	+	5.31E+02	3.70E+01	=
26	3.73E+03	6.90E+02	1.71E+03	1.45E+03	-	3.44E+03	1.71E+02	-	1.19E+03	3.08E+02	-	**8.19E+02**	**3.17E+02**	-	5.45E+03	2.59E+03	+	2.94E+03	5.98E+02	-	3.85E+03	7.26E+02	=
27	7.91E+02	1.09E+02	5.22E+02	7.76E+00	-	**5.11E+02**	**1.11E+01**	-	5.32E+02	1.48E+01	-	6.10E+02	5.77E+01	-	1.46E+03	1.70E+02	+	8.69E+02	1.76E+02	+	8.48E+02	1.34E+02	+
28	4.93E+02	2.54E+01	4.71E+02	2.15E+01	-	**4.60E+02**	**6.84E+00**	-	**4.60E+02**	**6.84E+00**	-	4.74E+02	2.21E+01	-	4.96E+02	1.79E+01	=	6.27E+02	4.82E+01	+	4.86E+02	2.30E+01	=
29	1.48E+03	2.63E+02	1.01E+03	2.25E+02	-	1.13E+03	1.15E+02	-	**6.84E+02**	**3.16E+02**	-	8.44E+02	2.01E+02	-	1.53E+03	2.11E+02	=	1.03E+03	2.48E+02	-	1.60E+03	3.87E+02	=
30	1.24E+06	4.72E+05	6.22E+05	3.53E+04	-	**6.01E+05**	**2.98E+04**	-	6.72E+05	6.96E+04	-	6.58E+06	5.27E+06	+	8.75E+05	8.54E+04	-	1.17E+06	3.16E+05	=	1.13E+06	3.50E+05	=

Table 15 Average error ± standard deviation and Wilcoxon signed-rank test (reference: cSNUM) for cSNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL and SNUM on CEC-2017 [2] in 100 dimensions.

Function	cSNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL			SNUM		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	2.24E+04	2.33E+04	4.52E+03	2.89E+03	-	3.92E+02	2.47E+02	-	4.31E+03	2.95E+03	-	**2.76E-07**	**5.88E-08**	-	3.58E+06	9.72E+05	+	1.22E+09	9.80E+08	+	2.70E+04	2.59E+04	=
2	1.29E+93	9.21E+93	3.08E+31	2.05E+32	+	6.81E+30	4.53E+31	=	9.44E+28	2.72E+29	+	**0.00E+00**	**0.00E+00**	-	1.86E+55	5.98E+55	+	2.05E+110	1.06E+111	+	9.32E+108	6.66E+109	+
3	1.85E+05	5.08E+04	1.26E+02	7.17E+01	-	5.20E+01	3.77E+01	-	**3.37E-02**	**2.89E-02**	-	1.42E+01	1.01E+02	-	1.57E+05	1.34E+04	-	2.77E+05	2.45E+04	+	5.99E+05	8.45E+04	+
4	2.05E+02	3.85E+01	2.12E+02	1.61E+01	=	2.11E+02	1.38E+01	=	2.20E+02	7.28E+00	+	**1.92E+02**	**3.68E+01**	=	2.84E+02	4.18E+01	+	7.20E+02	1.26E+02	+	2.29E+02	3.95E+01	+
5	6.16E+02	1.03E+02	3.39E+02	8.50E+01	-	6.52E+02	2.15E+01	+	**1.22E+01**	**2.99E+00**	-	2.11E+01	4.48E+00	-	5.21E+02	2.30E+01	-	3.50E+02	5.23E+01	-	6.53E+02	1.18E+02	=
6	3.23E-06	1.18E-06	1.67E-03	1.72E-04	+	2.78E-04	6.93E-04	+	2.42E-04	5.96E-05	+	7.95E-05	4.41E-04	-	4.22E+01	2.39E+00	+	1.97E+01	3.01E+00	+	**2.57E-07**	**1.82E-06**	-
7	5.97E+02	5.75E+01	4.87E+02	2.03E+02	-	7.87E+02	1.97E+01	+	**1.12E+02**	**1.54E+00**	-	1.28E+02	8.99E+00	-	7.46E+02	8.93E+01	+	7.88E+02	9.74E+01	+	6.94E+02	8.99E+01	+
8	6.25E+02	1.09E+02	3.67E+02	1.12E+02	-	6.50E+02	1.76E+01	+	**1.02E+01**	**2.98E+00**	-	2.11E+01	4.66E+00	-	5.84E+02	3.07E+01	-	3.78E+02	6.13E+01	-	6.95E+02	8.52E+01	+
9	2.58E+04	6.80E+03	1.76E-03	1.25E-02	-	4.59E-02	1.15E-01	-	1.41E-06	4.88E-07	-	**0.00E+00**	**0.00E+00**	-	1.56E+04	9.05E+02	-	2.11E+04	5.18E+03	-	2.93E+04	7.17E+03	+
10	1.22E+04	1.11E+03	1.78E+04	1.47E+03	+	2.82E+04	5.79E+02	+	1.66E+04	7.54E+03	=	**8.64E+03**	**2.73E+03**	-	1.15E+04	8.95E+02	-	2.95E+04	5.40E+02	+	1.27E+04	1.17E+03	+
11	9.02E+02	8.25E+02	5.52E+02	4.97E+01	=	2.44E+02	7.60E+01	-	**1.69E+02**	**5.82E+01**	-	1.23E+03	2.49E+02	+	1.23E+03	9.36E+01	+	1.08E+03	1.96E+02	+	3.70E+04	1.80E+04	+
12	2.37E+06	8.59E+05	3.70E+05	1.12E+05	-	5.55E+04	2.36E+04	-	**1.92E+04**	**7.50E+03**	-	7.20E+04	3.74E+05	-	1.11E+07	2.59E+06	+	3.89E+07	2.49E+07	+	4.97E+06	1.94E+06	+
13	9.62E+03	9.90E+03	3.89E+03	1.36E+03	-	6.12E+02	7.05E+01	-	**5.75E+02**	**9.03E+01**	-	6.15E+03	1.34E+03	=	2.38E+03	7.15E+02	-	1.45E+04	6.73E+03	+	8.92E+03	9.19E+03	=
14	5.08E+05	2.73E+05	4.66E+02	2.87E+01	-	2.88E+02	1.69E+01	-	**7.33E+01**	**1.24E+01**	-	6.22E+02	8.73E+01	-	4.55E+05	1.48E+05	=	2.53E+06	9.26E+05	+	2.01E+06	9.42E+05	+
15	5.23E+03	6.85E+03	6.83E+02	5.15E+01	-	2.24E+02	6.01E+01	-	**2.20E+02**	**5.22E+01**	-	1.44E+03	2.66E+02	-	5.67E+02	1.57E+02	-	3.87E+03	3.56E+03	=	9.37E+03	8.81E+03	+
16	4.48E+03	6.73E+02	5.23E+03	8.34E+02	+	6.08E+03	3.31E+02	+	**1.66E+03**	**4.94E+02**	-	1.71E+03	4.65E+02	-	3.21E+03	3.37E+02	-	2.65E+03	5.76E+02	-	4.82E+03	8.69E+02	+
17	3.60E+03	5.52E+02	3.91E+03	4.77E+02	+	4.02E+03	2.21E+02	+	3.51E+03	8.61E+02	=	**1.49E+03**	**2.73E+02**	-	2.63E+03	3.34E+02	-	2.25E+03	4.99E+02	-	3.71E+03	5.98E+02	=
18	6.76E+05	3.16E+05	1.18E+03	3.19E+02	-	1.80E+02	3.52E+01	-	**1.68E+02**	**3.85E+01**	-	7.08E+02	1.62E+02	-	8.74E+05	2.05E+05	+	5.90E+06	2.00E+06	+	2.34E+06	1.42E+06	+
19	8.39E+03	9.70E+03	3.03E+02	3.19E+01	-	1.26E+02	2.18E+01	-	**8.85E+01**	**1.23E+01**	-	8.18E+02	3.31E+02	-	5.47E+02	3.33E+02	-	4.16E+03	5.31E+03	-	1.01E+04	8.80E+03	=
20	3.29E+03	6.08E+02	3.89E+03	4.84E+02	+	4.33E+03	2.00E+02	+	**1.54E+03**	**4.68E+02**	-	2.02E+03	3.74E+02	-	2.35E+03	2.58E+02	-	4.55E+03	2.57E+02	+	3.50E+03	5.78E+02	=
21	8.80E+02	9.19E+01	5.98E+02	2.68E+02	-	8.78E+02	2.40E+01	=	1.03E+03	1.77E+01	+	**2.46E+02**	**5.20E+00**	-	1.06E+03	5.52E+01	+	6.05E+02	4.82E+01	-	9.34E+02	7.78E+01	+
22	1.30E+04	1.27E+03	2.05E+04	5.09E+03	+	2.92E+04	6.49E+02	+	1.00E+04	1.09E+03	-	**9.38E+03**	**3.49E+03**	-	1.38E+04	8.86E+02	+	3.01E+04	4.28E+03	+	1.43E+04	1.18E+03	+
23	8.95E+02	5.88E+01	1.00E+03	1.58E+02	+	1.18E+03	1.89E+01	+	1.17E+03	2.05E+01	+	**6.25E+02**	**6.96E+01**	-	2.09E+03	7.84E+01	+	1.17E+03	1.11E+02	+	9.22E+02	6.40E+01	+
24	1.47E+03	8.18E+01	1.64E+03	2.39E+01	+	1.48E+03	3.07E+01	=	1.64E+03	2.35E+01	+	**8.93E+02**	**1.41E+01**	-	1.89E+03	1.03E+02	+	2.02E+03	2.61E+02	+	1.52E+03	9.43E+01	+
25	7.47E+02	5.74E+01	7.25E+02	3.10E+01	-	7.36E+02	3.52E+01	=	7.10E+02	3.60E+01	-	**6.86E+02**	**4.37E+01**	-	8.02E+02	4.75E+01	+	1.34E+03	1.09E+02	+	7.72E+02	5.46E+01	+
26	9.87E+03	1.01E+03	8.94E+03	3.11E+03	=	8.83E+03	5.89E+02	-	3.17E+03	9.83E+01	-	**2.89E+03**	**5.68E+02**	-	1.56E+04	5.18E+03	+	1.09E+04	1.56E+03	+	1.04E+04	9.61E+02	+
27	8.71E+02	8.16E+01	5.88E+02	1.53E+01	-	**5.86E+02**	**2.16E+01**	-	5.93E+02	1.54E+01	-	6.73E+02	2.26E+01	-	1.35E+03	9.89E+01	+	1.19E+03	1.49E+02	+	8.35E+02	6.64E+01	-
28	5.50E+02	3.23E+01	5.42E+02	2.70E+01	=	5.40E+02	2.41E+01	=	5.29E+02	1.85E+01	-	**5.11E+02**	**6.37E+01**	-	6.36E+02	3.38E+01	+	1.53E+03	2.84E+02	+	5.56E+02	3.68E+01	=
29	3.78E+03	5.46E+02	3.56E+03	4.80E+02	=	4.26E+03	2.50E+02	+	**1.94E+03**	**6.82E+02**	-	2.06E+03	4.02E+02	-	3.94E+03	2.98E+02	+	3.49E+03	4.99E+02	-	5.00E+03	6.79E+02	+
30	9.01E+03	5.37E+03	3.76E+03	3.55E+02	-	**2.47E+03**	**1.46E+02**	-	2.60E+03	2.24E+02	-	2.87E+04	2.69E+04	+	4.36E+04	1.84E+04	+	8.61E+04	7.69E+04	+	1.42E+04	5.88E+03	+

D Extended numerical results for SNUM on CEC-2017 against population-based algorithms

Table 16 Average error ± standard deviation and Wilcoxon signed-rank test (reference: SNUM) for SNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 [2] in 10 dimensions.

Function	SNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	5.58E+03	4.36E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.54E-09	7.33E-09	-	2.41E+02	2.02E+02	-	2.10E+03	2.50E+03	-
2	1.61E+02	5.94E+02	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	2.16E-01	1.54E+00	-	3.92E-02	1.96E-01	-
3	4.03E+02	5.87E+02	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.14E-04	4.29E-04	-	1.94E+01	1.10E+01	-
4	1.32E+01	2.54E+01	1.24E-03	3.07E-03	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.90E+00	1.46E+00	-	3.97E+00	1.21E+00	+
5	2.17E+01	8.56E+00	**1.83E+00**	**1.60E+00**	-	9.31E+00	2.39E+00	-	5.39E+00	1.18E+00	-	2.13E+00	2.03E+00	-	1.81E+01	5.10E+00	=	1.63E+01	5.29E+00	-
6	6.57E-05	2.34E-04	7.58E-08	3.64E-08	+	3.57E-05	1.53E-05	+	0.00E+00	0.00E+00	-	1.34E-04	7.12E-04	+	9.18E-01	1.87E+00	+	8.40E-08	4.44E-07	+
7	3.20E+01	7.82E+00	1.16E+01	6.16E-01	-	2.27E+01	2.89E+00	-	1.43E+01	1.29E+00	-	1.11E+01	2.61E+00	-	1.78E+01	2.36E+00	-	3.22E+01	3.79E+00	=
8	2.50E+01	1.06E+01	1.68E+00	1.16E+00	-	1.04E+01	2.56E+00	-	4.26E+00	1.34E+00	-	3.22E+00	3.29E+00	-	9.95E+00	2.37E+00	-	2.01E+01	4.38E+00	-
9	3.72E+01	1.58E+02	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	-	2.80E-04	1.49E-03	-	8.91E-03	6.36E-02	-
10	6.86E+02	2.65E+02	3.19E+02	1.19E+02	-	7.01E+02	1.78E+02	=	2.86E+02	1.01E+02	-	5.39E+02	2.12E+02	-	5.03E+02	1.55E+02	-	1.14E+03	2.07E+02	+
11	1.86E+01	2.67E+01	1.67E+00	1.02E+00	-	3.22E+00	5.52E-01	-	1.19E-06	8.51E-06	-	9.03E-01	8.58E-01	-	1.73E+01	5.53E+00	+	4.49E+00	1.49E+00	-
12	1.51E+05	2.06E+05	9.82E+01	7.08E+01	-	2.44E+01	1.96E+01	-	1.19E+02	7.16E+01	-	2.19E+02	1.50E+02	-	4.76E+03	2.58E+03	-	1.38E+05	1.60E+05	=
13	7.96E+03	1.08E+04	4.62E+00	2.97E+00	-	7.74E+00	1.88E+00	-	4.94E+00	2.53E+00	-	8.80E+00	2.65E+01	-	1.46E+03	1.41E+03	-	3.20E+03	2.76E+03	=
14	1.03E+04	9.57E+03	3.67E+00	3.46E+00	-	7.18E+00	1.89E+00	-	3.32E+00	7.28E+00	-	2.52E+01	1.23E+01	-	6.29E+01	6.84E+01	-	9.46E+01	3.37E+01	-
15	1.18E+04	1.16E+04	2.69E-01	1.27E-01	-	6.38E-01	1.94E-01	-	4.08E-01	2.29E-01	-	3.14E+01	4.19E+01	-	1.87E+02	1.86E+02	-	2.14E+02	7.54E+01	-
16	2.17E+02	1.63E+02	3.42E+00	6.35E-01	-	5.41E+00	1.44E+00	-	1.41E+00	8.39E-01	-	1.49E+02	1.32E+02	-	8.36E+01	7.26E+01	-	1.23E+01	2.40E+01	-
17	5.84E+01	7.14E+01	1.21E+01	3.80E+00	-	3.44E+01	3.72E+00	=	**1.15E+01**	**6.34E+00**	-	6.61E+01	3.86E+01	+	2.47E+01	7.47E+00	=	4.65E+01	9.62E+00	+
18	2.08E+04	1.40E+04	1.41E+00	4.24E+00	-	**4.84E-01**	**2.04E-01**	-	5.96E+00	9.00E+00	-	4.43E+01	5.10E+01	-	2.26E+03	2.19E+03	-	9.41E+03	6.52E+03	-
19	1.00E+04	1.09E+04	**2.23E-01**	**1.37E-01**	-	1.19E+00	3.34E-01	-	2.35E-01	3.79E-01	-	3.63E+00	3.65E+00	-	1.48E+02	2.38E+02	-	1.24E+02	1.08E+02	-
20	1.07E+01	9.14E+00	**1.74E+00**	**2.03E+00**	-	2.15E+01	3.97E+00	+	2.71E+00	6.51E+00	-	1.23E+02	7.83E+01	+	2.98E+01	1.09E+01	+	2.26E+01	9.74E+00	+
21	2.21E+02	4.63E+01	1.39E+02	3.97E+01	-	1.40E+02	5.13E+01	-	**1.02E+02**	**1.01E+01**	-	1.55E+02	5.24E+01	-	1.05E+02	2.15E+01	-	1.48E+02	5.46E+01	-
22	7.52E+02	5.36E+02	9.85E+01	1.10E+01	-	1.00E+02	1.80E-01	-	1.00E+02	1.13E-01	-	9.93E+01	5.57E+00	-	9.68E+01	1.68E+01	-	**9.37E+01**	**2.22E+01**	-
23	3.31E+02	2.44E+01	3.04E+02	2.01E+00	-	3.07E+02	2.15E+00	-	3.06E+02	1.24E+00	-	**2.87E+02**	**5.52E+01**	-	3.42E+02	1.05E+01	+	3.11E+02	6.41E+00	-
24	3.40E+02	8.58E+01	**2.14E+02**	**8.48E+01**	-	3.28E+02	3.57E+01	-	2.90E+02	8.55E+01	-	2.33E+02	1.10E+02	-	2.27E+02	1.35E+02	-	3.14E+02	6.72E+01	-
25	4.21E+02	6.10E+01	4.13E+02	2.10E+01	-	4.06E+02	1.75E+01	-	4.28E+02	2.16E+01	-	4.07E+02	6.65E+01	-	**4.04E+02**	**1.45E+01**	-	4.26E+02	2.25E+01	-
26	1.02E+03	5.72E+02	**2.66E+02**	**4.67E+01**	-	3.00E+02	0.00E+00	-	3.00E+02	0.00E+00	-	2.85E+02	1.45E+02	-	2.67E+02	7.66E+01	-	3.02E+02	4.66E+01	-
27	4.37E+02	3.53E+01	3.92E+02	2.40E+00	-	**3.89E+02**	**2.22E-01**	-	3.90E+02	9.32E-01	-	3.95E+02	3.98E+00	-	4.27E+02	1.35E+01	=	3.93E+02	3.35E+00	-
28	5.67E+02	1.29E+02	3.22E+02	9.83E+01	-	3.43E+02	1.00E+02	-	3.17E+02	6.97E+01	-	4.18E+02	1.71E+02	-	**2.94E+02**	**4.21E+01**	-	4.49E+02	1.56E+02	-
29	3.76E+02	8.02E+01	**2.45E+02**	**5.82E+00**	-	2.57E+02	6.87E+00	-	2.48E+02	6.52E+00	-	2.87E+02	5.30E+01	-	2.78E+02	1.36E+01	-	2.87E+02	1.63E+01	-
30	5.02E+05	6.81E+05	4.09E+02	1.82E+01	-	**3.95E+02**	**1.01E-01**	-	4.30E+02	2.59E+01	-	2.29E+03	1.04E+04	-	7.09E+03	3.12E+03	-	2.98E+05	5.08E+05	-

Table 17 Average error ± standard deviation and Wilcoxon signed-rank test (reference: SNUM) for SNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 [2] in 30 dimensions.

Function	SNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	9.53E+03	7.62E+03	1.25E-04	8.32E-05	-	3.68E-06	1.95E-06	-	**0.00E+00**	**0.00E+00**	-	6.27E-08	1.93E-08	-	7.55E+02	6.11E+02	-	3.66E+03	3.76E+03	-
2	3.69E+23	2.41E+24	4.45E+00	1.40E+01	-	**1.72E-08**	**6.48E-08**	-	1.96E-02	1.40E-01	-	1.17E+02	8.35E+02	-	9.50E+06	1.71E+07	-	8.52E+16	5.81E+17	=
3	1.04E+05	3.83E+04	3.59E+01	2.40E+02	-	4.03E-08	1.98E-08	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	4.08E+02	2.03E+02	-	2.08E+04	4.64E+03	-
4	7.37E+01	2.73E+01	6.93E+01	1.17E+01	-	5.87E+01	7.78E-01	-	5.91E+01	3.45E+00	-	**5.53E+01**	**1.65E+01**	-	5.88E+01	3.40E+01	-	9.49E+01	2.35E+01	+
5	1.16E+02	3.45E+01	1.17E+01	3.96E+00	-	1.25E+02	1.07E+01	+	3.03E+01	4.32E+01	-	**4.00E+00**	**1.94E+00**	-	1.12E+02	1.33E+01	=	7.57E+01	4.35E+01	-
6	6.68E-05	3.34E-04	4.60E-05	1.42E-05	+	2.92E-04	5.66E-05	+	**1.31E-07**	**9.11E-08**	+	5.40E-05	3.78E-04	+	2.16E+01	3.97E+00	+	4.87E-01	4.24E-01	+
7	1.36E+02	2.41E+01	3.94E+01	1.23E+01	-	1.70E+02	1.33E+01	+	4.50E+01	2.47E+01	-	**3.52E+01**	**1.81E+00**	-	1.35E+02	1.62E+01	=	1.89E+02	2.09E+01	+
8	1.34E+02	3.45E+01	2.39E+01	3.50E+01	-	1.25E+02	1.23E+01	=	6.99E+00	1.93E+01	-	**4.35E+00**	**2.66E+00**	-	8.10E+01	1.04E+01	-	7.14E+01	4.37E+01	-
9	2.62E+03	1.93E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.38E+03	2.72E+02	-	3.46E+01	2.73E+01	-
10	3.22E+03	5.03E+02	4.40E+03	7.91E+02	+	6.19E+03	3.00E+02	+	5.28E+03	8.28E+02	+	**1.87E+03**	**7.56E+02**	-	3.13E+03	3.46E+02	=	6.99E+03	2.62E+02	+
11	1.15E+02	4.81E+01	4.40E+01	1.78E+01	-	3.19E+01	1.35E+01	-	**1.78E+01**	**2.48E+01**	-	6.32E+01	3.96E+01	-	8.83E+01	2.12E+01	-	8.99E+01	4.19E+01	-
12	1.31E+06	1.09E+06	2.53E+03	1.27E+03	-	**5.78E+02**	**3.04E+02**	-	6.15E+02	3.13E+02	-	1.38E+03	3.62E+02	-	9.51E+04	5.13E+04	-	1.04E+05	1.70E+05	-
13	2.45E+04	2.55E+04	8.02E+01	1.96E+01	-	6.68E+01	8.38E+00	-	**2.90E+01**	**2.42E+01**	-	9.97E+02	9.09E+02	-	3.21E+03	2.88E+03	-	2.06E+04	1.78E+04	=
14	9.20E+05	1.07E+06	6.36E+01	3.95E+00	-	4.86E+01	5.59E+00	-	**2.61E+01**	**8.63E+00**	-	1.74E+02	4.94E+01	-	3.48E+03	2.63E+03	-	1.13E+04	8.70E+03	-
15	2.12E+04	1.46E+04	3.75E+01	5.32E+00	-	2.36E+01	3.04E+00	-	**6.51E+00**	**2.41E+00**	-	2.75E+02	1.57E+02	-	2.13E+03	1.63E+03	-	2.90E+04	2.57E+04	=
16	1.24E+03	3.53E+02	9.04E+02	2.05E+02	-	8.20E+02	1.79E+02	-	**4.75E+02**	**3.40E+02**	-	5.83E+02	2.48E+02	-	8.46E+02	1.53E+02	-	7.48E+02	4.33E+02	-
17	7.61E+02	2.49E+02	1.78E+02	3.36E+01	-	2.14E+02	2.57E+01	-	1.88E+02	4.84E+01	-	2.48E+02	1.60E+02	-	3.31E+02	1.13E+02	-	**1.52E+02**	**7.04E+01**	-
18	3.14E+06	3.50E+06	4.19E+01	4.31E+00	-	3.05E+01	1.84E+00	-	**2.39E+01**	**1.92E+00**	-	2.20E+02	1.23E+02	-	8.77E+04	3.31E+04	-	5.31E+05	2.20E+05	-
19	1.67E+04	1.74E+04	2.26E+01	2.14E+00	-	2.41E+01	1.70E+00	-	**1.08E+01**	**2.94E+00**	-	1.91E+02	7.37E+01	-	1.71E+03	1.69E+03	-	1.23E+04	1.13E+04	=
20	7.31E+02	2.96E+02	2.64E+02	5.36E+01	-	3.12E+02	4.74E+01	-	**1.56E+02**	**1.01E+02**	-	4.97E+02	2.03E+02	-	3.58E+02	1.17E+02	-	3.33E+02	1.37E+02	-
21	3.39E+02	3.55E+01	2.16E+02	1.59E+01	-	3.19E+02	9.95E+00	-	2.83E+02	5.58E+01	-	**2.13E+02**	**4.75E+00**	-	3.05E+02	3.30E+01	-	2.65E+02	4.14E+01	-
22	3.49E+03	1.04E+03	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	1.09E+03	1.12E+03	-	**1.00E+02**	**2.36E-03**	-	1.01E+02	1.94E+00	-
23	4.78E+02	2.91E+01	3.80E+02	2.38E+01	-	4.67E+02	1.08E+01	=	4.55E+02	1.47E+01	-	**3.47E+02**	**5.02E+01**	-	6.81E+02	3.79E+01	+	3.97E+02	1.62E+01	-
24	7.03E+02	1.11E+02	4.54E+02	3.48E+01	-	5.31E+02	8.43E+00	-	5.41E+02	2.90E+01	-	**4.23E+02**	**3.24E+00**	-	7.39E+02	4.57E+01	+	4.72E+02	2.00E+01	-
25	3.98E+02	2.12E+01	3.87E+02	2.14E-02	-	3.87E+02	7.72E-03	-	3.87E+02	1.06E-02	-	3.87E+02	1.49E-02	-	**3.86E+02**	**1.77E+00**	-	4.02E+02	1.76E+01	+
26	2.37E+03	6.26E+02	7.75E+02	3.46E+02	-	2.00E+03	9.68E+01	-	1.09E+03	3.17E+02	-	**5.09E+02**	**2.71E+02**	-	2.04E+03	1.73E+03	=	1.43E+03	4.62E+02	-
27	5.42E+02	1.75E+01	5.02E+02	4.03E+00	-	**4.99E+02**	**7.99E+00**	-	5.05E+02	6.37E+00	-	5.20E+02	1.32E+01	-	7.08E+02	5.41E+01	+	5.32E+02	2.07E+01	-
28	4.31E+02	4.04E+01	3.61E+02	4.43E+01	-	3.13E+02	3.36E+01	-	3.14E+02	3.52E+01	-	**3.13E+02**	**3.48E+01**	-	3.91E+02	1.77E+01	-	4.35E+02	2.69E+01	=
29	1.11E+03	2.58E+02	7.33E+02	8.42E+01	-	7.51E+02	3.73E+01	-	**4.89E+02**	**7.49E+01**	-	5.68E+02	1.54E+02	-	7.81E+02	1.21E+02	-	6.26E+02	9.19E+01	-
30	1.60E+04	1.13E+04	2.04E+03	6.20E+01	-	**1.98E+03**	**1.88E+01**	-	2.01E+03	6.24E+01	-	3.07E+03	1.93E+03	-	5.69E+03	1.94E+03	-	3.65E+04	3.88E+04	+

Table 18 Average error ± standard deviation and Wilcoxon signed-rank test (reference: SNUM) for SNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 [2] in 50 dimensions.

Function	SNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.34E+04	1.25E+04	7.30E+02	4.56E+02	-	2.44E-01	2.97E-01	-	2.57E-02	2.86E-02	-	**1.23E-07**	**4.41E-08**	-	4.97E+03	2.50E+03	-	1.02E+06	3.03E+06	+
2	6.62E+35	4.72E+36	1.25E+04	4.86E+04	-	3.77E+04	1.96E+05	-	**1.87E+02**	**5.84E+02**	-	2.77E+05	1.98E+06	-	1.25E+18	4.36E+18	-	1.57E+39	1.01E+40	+
3	2.24E+05	5.44E+04	3.51E+03	1.04E+04	-	2.18E-04	1.56E-04	-	3.57E-08	3.23E-08	-	**0.00E+00**	**0.00E+00**	-	1.54E+04	3.23E+03	-	7.02E+04	9.11E+03	-
4	1.14E+02	5.90E+01	6.83E+01	4.60E+01	-	8.19E+01	5.67E+01	-	7.53E+01	5.51E+01	-	**3.82E+01**	**4.42E+01**	-	1.11E+02	4.11E+01	=	1.98E+02	4.63E+01	+
5	2.29E+02	4.90E+01	1.11E+02	1.04E+02	-	2.72E+02	1.16E+01	+	**5.99E+00**	**2.02E+00**	-	7.47E+00	2.74E+00	-	2.01E+02	1.37E+01	-	1.01E+02	1.80E+01	-
6	4.23E-05	2.12E-04	2.24E-04	2.35E-05	+	9.82E-05	2.94E-05	+	1.46E-05	6.24E-06	+	**1.89E-07**	**1.72E-07**	+	3.37E+01	3.92E+00	+	4.52E+00	1.70E+00	+
7	2.58E+02	3.91E+01	1.22E+02	5.00E+01	-	3.39E+02	1.56E+01	+	6.08E+01	2.56E+01	-	**5.93E+01**	**2.09E+00**	-	2.78E+02	3.42E+01	+	2.60E+02	1.04E+02	=
8	2.40E+02	5.43E+01	1.46E+02	1.18E+02	-	2.71E+02	1.76E+01	+	**5.81E+00**	**1.88E+00**	-	7.94E+00	2.54E+00	-	1.99E+02	1.52E+01	-	9.73E+01	1.54E+01	-
9	6.60E+03	3.91E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	6.13E+03	7.42E+02	=	1.73E+03	1.19E+03	-
10	5.25E+03	7.71E+02	7.70E+03	9.00E+02	+	1.21E+04	3.37E+02	+	9.37E+03	3.57E+03	+	**3.00E+03**	**9.51E+02**	-	5.20E+03	5.52E+02	=	1.29E+04	4.00E+02	+
11	1.62E+03	2.43E+03	1.32E+02	9.01E+00	-	9.67E+01	8.37E+00	-	**3.24E+01**	**4.63E+00**	-	2.13E+02	5.70E+01	-	1.29E+02	1.43E+01	-	1.74E+02	4.91E+01	-
12	2.62E+06	1.48E+06	2.97E+04	1.34E+04	-	1.75E+03	5.18E+02	-	**1.60E+03**	**4.19E+02**	-	3.89E+06	2.76E+07	-	1.01E+06	3.45E+05	-	1.30E+06	1.05E+06	-
13	1.65E+04	1.42E+04	3.04E+02	3.78E+01	-	1.94E+02	3.40E+01	-	**1.57E+02**	**4.53E+01**	-	2.32E+03	1.60E+03	-	9.01E+02	5.62E+02	-	8.40E+03	5.19E+03	=
14	7.60E+05	5.23E+05	1.36E+02	9.33E+00	-	1.01E+02	7.72E+00	-	**3.11E+01**	**4.86E+00**	-	2.93E+02	8.35E+01	-	2.60E+04	1.34E+04	-	1.25E+05	6.54E+04	-
15	1.48E+04	8.95E+03	1.27E+02	1.27E+01	-	8.36E+01	8.41E+00	-	**3.08E+01**	**5.59E+00**	-	8.25E+02	2.08E+02	-	1.20E+03	7.81E+02	-	6.97E+03	6.03E+03	-
16	2.27E+03	4.95E+02	1.93E+03	3.90E+02	-	1.96E+03	1.98E+02	-	1.02E+03	6.77E+02	-	**9.44E+02**	**3.40E+02**	-	1.25E+03	2.31E+02	-	9.78E+02	3.64E+02	-
17	1.59E+03	3.82E+02	1.47E+03	2.16E+02	=	1.32E+03	1.29E+02	-	**6.34E+02**	**4.50E+02**	-	8.46E+02	2.34E+02	-	1.03E+03	1.54E+02	-	1.16E+03	4.90E+02	-
18	2.78E+06	2.04E+06	1.17E+02	1.73E+01	-	5.39E+01	5.61E+00	-	**3.31E+01**	**6.10E+00**	-	3.87E+02	1.45E+02	-	3.36E+05	1.46E+05	-	1.74E+06	9.40E+05	-
19	1.94E+04	1.70E+04	7.10E+01	5.98E+00	-	5.07E+01	4.64E+00	-	**2.42E+01**	**6.98E+00**	-	2.03E+02	5.75E+01	-	8.67E+03	3.91E+03	-	1.45E+04	9.40E+03	=
20	1.40E+03	3.69E+02	1.34E+03	2.35E+02	=	1.16E+03	1.22E+02	-	**3.32E+02**	**3.33E+02**	-	9.05E+02	2.79E+02	-	7.80E+02	1.97E+02	-	1.31E+03	2.81E+02	=
21	4.62E+02	4.92E+01	3.17E+02	1.16E+02	-	4.73E+02	1.16E+01	=	4.75E+02	5.98E+01	+	**2.12E+02**	**3.57E+00**	-	4.33E+02	2.14E+01	-	2.93E+02	3.32E+01	-
22	5.98E+03	6.73E+02	**2.14E+03**	**3.56E+03**	-	7.55E+03	5.81E+03	+	2.44E+03	4.09E+03	-	3.48E+03	1.92E+03	-	5.98E+03	9.89E+02	=	6.99E+03	6.55E+03	=
23	7.13E+02	5.01E+01	6.84E+02	1.02E+02	=	6.89E+02	1.78E+01	-	7.16E+02	2.78E+01	=	**4.33E+02**	**1.41E+01**	-	1.07E+03	7.19E+01	+	5.67E+02	3.41E+01	-
24	1.26E+03	1.86E+02	7.42E+02	1.22E+02	-	7.43E+02	1.87E+01	-	8.09E+02	2.32E+01	-	**4.95E+02**	**6.45E+00**	-	1.08E+03	7.07E+01	-	6.78E+02	4.67E+01	-
25	5.31E+02	3.70E+01	4.90E+02	2.67E+01	-	4.81E+02	2.80E+00	-	**4.81E+02**	**2.32E+00**	-	4.85E+02	1.71E+01	-	5.55E+02	2.65E+01	+	6.21E+02	2.61E+01	+
26	3.85E+03	7.26E+02	1.71E+03	1.45E+03	-	3.44E+03	1.71E+02	-	1.19E+03	3.08E+02	-	**8.19E+02**	**3.17E+02**	-	5.45E+03	2.59E+03	+	2.94E+03	5.98E+02	-
27	8.48E+02	1.34E+02	5.22E+02	7.76E+00	-	**5.11E+02**	**1.11E+01**	-	5.32E+02	1.48E+01	-	6.10E+02	5.77E+01	-	1.46E+03	1.70E+02	+	8.69E+02	1.76E+02	=
28	4.86E+02	2.30E+01	4.71E+02	2.15E+01	-	**4.60E+02**	**6.84E+00**	-	**4.60E+02**	**6.84E+00**	-	4.74E+02	2.21E+01	-	4.96E+02	1.79E+01	+	6.27E+02	4.82E+01	+
29	1.60E+03	3.87E+02	1.01E+03	2.25E+02	-	1.13E+03	1.15E+02	-	**6.84E+02**	**3.16E+02**	-	8.44E+02	2.01E+02	-	1.53E+03	2.11E+02	=	1.03E+03	2.48E+02	-
30	1.13E+06	3.50E+05	6.22E+05	3.53E+04	-	**6.01E+05**	**2.98E+04**	-	6.72E+05	6.96E+04	-	6.58E+06	5.27E+06	+	8.75E+05	8.54E+04	-	1.17E+06	3.16E+05	=

Table 19 Average error ± standard deviation and Wilcoxon signed-rank test (reference: SNUM) for SNUM against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 [2] in 100 dimensions.

Function	SNUM		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	2.70E+04	2.59E+04	4.52E+03	2.89E+03	-	3.92E+02	2.47E+02	-	4.31E+03	2.95E+03	-	**2.76E-07**	**5.88E-08**	-	3.58E+06	9.72E+05	+	1.22E+09	9.80E+08	+
2	9.32E+108	6.66E+109	3.08E+31	2.05E+32	-	6.81E+30	4.53E+31	-	9.44E+28	2.72E+29	-	**0.00E+00**	**0.00E+00**	-	1.86E+55	5.98E+55	-	2.05E+110	1.06E+111	+
3	5.99E+05	8.45E+04	1.26E+02	7.17E+01	-	5.20E+01	3.77E+01	-	**3.37E-02**	**2.89E-02**	-	1.42E+01	1.01E+02	-	1.57E+05	1.34E+04	-	2.77E+05	2.45E+04	-
4	2.29E+02	3.95E+01	2.12E+02	1.61E+01	-	2.11E+02	1.38E+01	-	2.20E+02	7.28E+00	=	**1.92E+02**	**3.68E+01**	-	2.84E+02	4.18E+01	+	7.20E+02	1.26E+02	+
5	6.53E+02	1.18E+02	3.39E+02	8.50E+01	-	6.52E+02	2.15E+01	=	**1.22E+01**	**2.99E+00**	-	2.11E+01	4.48E+00	-	5.21E+02	2.30E+01	-	3.50E+02	5.23E+01	-
6	**2.57E-07**	**1.82E-06**	1.67E-03	1.72E-04	+	2.78E-04	6.93E-04	+	2.42E-04	5.96E-05	+	7.95E-05	4.41E-04	+	4.22E+01	2.39E+00	+	1.97E+01	3.01E+00	+
7	6.94E+02	8.99E+01	4.87E+02	2.03E+02	-	7.87E+02	1.97E+01	+	**1.12E+02**	**1.54E+00**	-	1.28E+02	8.99E+00	-	7.46E+02	8.93E+01	+	7.88E+02	9.74E+01	+
8	6.95E+02	8.52E+01	3.67E+02	1.12E+02	-	6.50E+02	1.76E+01	-	**1.02E+01**	**2.98E+00**	-	2.11E+01	4.66E+00	-	5.84E+02	3.07E+01	-	3.78E+02	6.13E+01	-
9	2.93E+04	7.17E+03	1.76E-03	1.25E-02	-	4.59E-02	1.15E-01	-	1.41E-06	4.88E-07	-	**0.00E+00**	**0.00E+00**	-	1.56E+04	9.05E+02	-	2.11E+04	5.18E+03	-
10	1.27E+04	1.17E+03	1.78E+04	1.47E+03	+	2.82E+04	5.79E+02	+	1.66E+04	7.54E+03	=	**8.64E+03**	**2.73E+03**	-	1.15E+04	8.95E+02	-	2.95E+04	5.40E+02	+
11	3.70E+04	1.80E+04	5.52E+02	4.97E+01	-	2.44E+02	7.60E+01	-	**1.69E+02**	**5.82E+01**	-	1.23E+03	2.49E+02	-	1.23E+03	9.36E+01	-	1.08E+03	1.96E+02	-
12	4.97E+06	1.94E+06	3.70E+05	1.12E+05	-	5.55E+04	2.36E+04	-	**1.92E+04**	**7.50E+03**	-	7.20E+04	3.74E+05	-	1.11E+07	2.59E+06	+	3.89E+07	2.49E+07	+
13	8.92E+03	9.19E+03	3.89E+03	1.36E+03	=	6.12E+02	7.05E+01	-	**5.75E+02**	**9.03E+01**	-	6.15E+03	1.34E+03	=	2.38E+03	7.15E+02	-	1.45E+04	6.73E+03	+
14	2.01E+06	9.42E+05	4.66E+02	2.87E+01	-	2.88E+02	1.69E+01	-	**7.33E+01**	**1.24E+01**	-	6.22E+02	8.73E+01	-	4.55E+05	1.48E+05	-	2.53E+06	9.26E+05	+
15	9.37E+03	8.81E+03	6.83E+02	5.15E+01	-	2.24E+02	6.01E+01	-	**2.20E+02**	**5.22E+01**	-	1.44E+03	2.66E+02	-	5.67E+02	1.57E+02	-	3.87E+03	3.56E+03	-
16	4.82E+03	8.69E+02	5.23E+03	8.34E+02	+	6.08E+03	3.31E+02	+	**1.66E+03**	**4.94E+02**	-	1.71E+03	4.65E+02	-	3.21E+03	3.37E+02	-	2.65E+03	5.76E+02	-
17	3.71E+03	5.98E+02	3.91E+03	4.77E+02	+	4.02E+03	2.21E+02	+	3.51E+03	8.61E+02	=	**1.49E+03**	**2.73E+02**	-	2.63E+03	3.34E+02	-	2.25E+03	4.99E+02	-
18	2.34E+06	1.42E+06	1.18E+03	3.19E+02	-	1.80E+02	3.52E+01	-	**1.68E+02**	**3.85E+01**	-	7.08E+02	1.62E+02	-	8.74E+05	2.05E+05	-	5.90E+06	2.00E+06	+
19	1.01E+04	8.80E+03	3.03E+02	3.19E+01	-	1.26E+02	2.18E+01	-	**8.85E+01**	**1.23E+01**	-	8.18E+02	3.31E+02	-	5.47E+02	3.33E+02	-	4.16E+03	5.31E+03	-
20	3.50E+03	5.78E+02	3.89E+03	4.84E+02	+	4.33E+03	2.00E+02	+	**1.54E+03**	**4.68E+02**	-	2.02E+03	3.74E+02	-	2.35E+03	2.58E+02	-	4.55E+03	2.57E+02	+
21	9.34E+02	7.78E+01	5.98E+02	2.68E+02	-	8.78E+02	2.40E+01	-	1.03E+03	1.77E+01	+	**2.46E+02**	**5.20E+00**	-	1.06E+03	5.52E+01	+	6.05E+02	4.82E+01	-
22	1.43E+04	1.18E+03	2.05E+04	5.09E+03	+	2.92E+04	6.49E+02	+	1.00E+04	1.09E+03	-	**9.38E+03**	**3.49E+03**	-	1.38E+04	8.86E+02	=	3.01E+04	4.28E+03	+
23	9.22E+02	6.40E+01	1.00E+03	1.58E+02	+	1.18E+03	1.89E+01	+	1.17E+03	2.05E+01	+	**6.25E+02**	**6.96E+01**	-	2.09E+03	7.84E+01	+	1.17E+03	1.11E+02	+
24	1.52E+03	9.43E+01	1.64E+03	2.39E+01	+	1.48E+03	3.07E+01	-	1.64E+03	2.35E+01	+	**8.93E+02**	**1.41E+01**	-	1.89E+03	1.03E+02	+	2.02E+03	2.61E+02	+
25	7.72E+02	5.46E+01	7.25E+02	3.10E+01	-	7.36E+02	3.52E+01	-	7.10E+02	3.60E+01	-	**6.86E+02**	**4.37E+01**	-	8.02E+02	4.75E+01	+	1.34E+03	1.09E+02	+
26	1.04E+04	9.61E+02	8.94E+03	3.11E+03	=	8.83E+03	5.89E+02	-	3.17E+03	9.83E+01	-	**2.89E+03**	**5.68E+02**	-	1.56E+04	5.18E+03	+	1.09E+04	1.56E+03	=
27	8.35E+02	6.64E+01	5.88E+02	1.53E+01	-	**5.86E+02**	**2.16E+01**	-	5.93E+02	1.54E+01	-	6.73E+02	2.26E+01	-	1.35E+03	9.89E+01	+	1.19E+03	1.49E+02	+
28	5.56E+02	3.68E+01	5.42E+02	2.70E+01	=	5.40E+02	2.41E+01	-	5.29E+02	1.85E+01	-	**5.11E+02**	**6.37E+01**	-	6.36E+02	3.38E+01	+	1.53E+03	2.84E+02	+
29	5.00E+03	6.79E+02	3.56E+03	4.80E+02	-	4.26E+03	2.50E+02	-	**1.94E+03**	**6.82E+02**	-	2.06E+03	4.02E+02	-	3.94E+03	2.98E+02	-	3.49E+03	4.99E+02	-
30	1.42E+04	5.88E+03	3.76E+03	3.55E+02	-	**2.47E+03**	**1.46E+02**	-	2.60E+03	2.24E+02	-	2.87E+04	2.69E+04	+	4.36E+04	1.84E+04	+	8.61E+04	7.69E+04	+

E Extended numerical results for SNUM and cSNUM on CEC-2017 against population-based algorithms (Holm-Bonferroni procedure)

Table 20 Holm-Bonferroni procedure (Reference: EBOwithCMAR , Rank = 7.13E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions on D 10. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	LSHADE_SPACMA	6.77E+00	-5.80E-01	2.81E-01	5.00E-02	Not rejected
2	JSO	5.77E+00	-2.16E+00	1.54E-02	2.50E-02	Rejected
3	RB_IPOP_CMA_ES	5.23E+00	-3.00E+00	1.33E-03	1.67E-02	Rejected
4	PPSO	4.27E+00	-4.53E+00	2.91E-06	1.25E-02	Rejected
5	TLBO_FL	3.33E+00	-6.01E+00	9.37E-10	1.00E-02	Rejected
6	cSNUM	2.80E+00	-6.85E+00	3.65E-12	8.33E-03	Rejected
7	SNUM	1.53E+00	-8.85E+00	4.21E-19	7.14E-03	Rejected

Table 21 Holm-Bonferroni procedure (Reference: LSHADE_SPACMA , Rank = 6.80E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions on D 30. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	6.23E+00	-8.96E-01	1.85E-01	5.00E-02	Not rejected
2	EBOwithCMAR	5.97E+00	-1.32E+00	9.38E-02	2.50E-02	Not rejected
3	JSO	5.53E+00	-2.00E+00	2.26E-02	1.67E-02	Not rejected
4	PPSO	3.73E+00	-4.85E+00	6.21E-07	1.25E-02	Rejected
5	TLBO_FL	3.40E+00	-5.38E+00	3.81E-08	1.00E-02	Rejected
6	cSNUM	3.13E+00	-5.80E+00	3.37E-09	8.33E-03	Rejected
7	SNUM	1.67E+00	-8.12E+00	2.40E-16	7.14E-03	Rejected

Table 22 Holm-Bonferroni procedure (Reference: LSHADE_SPACMA , Rank = 6.77E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions on D 50. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	6.17E+00	-9.49E-01	1.71E-01	5.00E-02	Not rejected
2	EBOwithCMAR	5.60E+00	-1.84E+00	3.25E-02	2.50E-02	Not rejected
3	JSO	5.07E+00	-2.69E+00	3.59E-03	1.67E-02	Rejected
4	PPSO	3.63E+00	-4.95E+00	3.63E-07	1.25E-02	Rejected
5	TLBO_FL	3.43E+00	-5.27E+00	6.80E-08	1.00E-02	Rejected
6	cSNUM	3.37E+00	-5.38E+00	3.81E-08	8.33E-03	Rejected
7	SNUM	2.20E+00	-7.22E+00	2.59E-13	7.14E-03	Rejected

Table 23 Holm-Bonferroni procedure (Reference: LSHADE_SPACMA , Rank = 6.67E+00) for the eight algorithms under consideration over the 30 CEC 2017 benchmark functions D 100. Higher ranks indicate better algorithms.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	6.63E+00	-5.27E-02	4.79E-01	5.00E-02	Not rejected
2	EBOwithCMAR	4.93E+00	-2.74E+00	3.07E-03	2.50E-02	Rejected
3	JSO	4.83E+00	-2.90E+00	1.87E-03	1.67E-02	Rejected
4	cSNUM	4.20E+00	-3.90E+00	4.81E-05	1.25E-02	Rejected
5	PPSO	3.33E+00	-5.27E+00	6.80E-08	1.00E-02	Rejected
6	SNUM	2.93E+00	-5.90E+00	1.79E-09	8.33E-03	Rejected
7	TLBO_FL	2.47E+00	-6.64E+00	1.56E-11	7.14E-03	Rejected