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

A single-solution–compact hybrid algorithm for continuous optimization

Souheila Khalfi · **Giovanni Iacca**[⋆] · **Amer Draa**

October 2, 2022

Abstract This research paper proposes a memetic algorithm based on a hybridization of two metaheuristic approaches, a single-solution method and a compact optimization algorithm. The hybrid algorithm is thus a bi-module framework, where each module encapsulates a different search logic. Both modules use the Non-Uniform Mutation, although with different flavors: the first one acting on a single variable at a time, the second one acting on multiple variables. Hence, the algorithm is dubbed "compact Single/Multi Non-Uniform Mutation" (in short, cSM). It is designed for being suitable for tackling optimization problems on memory-constrained devices, i.e., devices for which the available memory may be not enough to run population-based metaheuristics. The performance of cSM is evaluated by an extensive comparative analysis including 12 state-of-the-art memory-saving (also called "lightweight") algorithms on three well-known testbeds, namely the BBOB, the CEC-2014, and CEC-2017 benchmarks, as well as seven real-world optimization problems included in the CEC-2011 benchmark. In the case of the CEC benchmarks, our method is also compared against the top (population-based) algorithms that participated in respective competitions. The numerical results indicate that, compared to all the other lightweight algorithms under study, the proposed algorithm is better at handling most functions at different dimensionalities, especially in the case of non-separable problems.

Keywords Memetic Computing · Single-Solution Metaheuristic · Compact optimization · Non-Uniform Mutation · Limited-Memory Hardware.

1 Introduction

Among the various optimization techniques available in the literature, metaheuristics are known to be applicable to different types of optimization problems where exact algorithms may be ineffective

[⋆] Corresponding author

Souheila Khalfi
Department of Fundamental Informatics and its Applications, Constantine 2 University, Algeria
Department of Mathematics and Computer Science, Mila University Center, Algeria
E-mail: souheila.khalfi@gmail.com, souheila.khalfi@univ-constantine2.dz

Giovanni Iacca
Department of Information Engineering and Computer Science, University of Trento, Italy
E-mail: giovanni.iacca@unitn.it

Amer Draa
Department of Fundamental Informatics and its Applications, Constantine 2 University, Algeria
E-mail: draa.amer@gmail.com

or inapplicable. While many taxonomies can be used to categorize metaheuristics, one possible way to divide them is based on how many solutions are manipulated during the search process. From this point of view, metaheuristics can be roughly categorized into *population-based* approaches (Boussaïd et al, 2013) and *single-solution algorithms*—sometimes referred to as solo-search algorithms.

In the first case, multiple solutions (a "population") are handled. This category represents nowadays the main trend in the field of metaheuristics, due to the fact that population-based algorithms are effective at exploring complex, large search spaces. Some recent examples of population-based algorithms can be found for instance in (Abualigah et al, 2021; Agushaka et al, 2022; Oyelade et al, 2022).

In the second case, on the other hand, only one solution is manipulated during the search. This approach may be especially appropriate in some specific applications plagued by severe hardware resource limitations, e.g. due to the use of embedded systems. In these contexts, memory represents an especially important aspect. In fact, increasing the available memory often increases packaging and cooling costs, as well as the physical size and the energy consumption of the entire device. Consequently, choosing an optimizer with a low memory consumption can be crucial in these situations.

The use of *lightweight* algorithms (Khalfi et al, 2022) is a viable way to perform optimization in these scenarios. Besides the aforementioned *single-solution algorithms*, an alternative subclass of lightweight algorithms is represented by the so-called *compact optimization* algorithms (Neri et al, 2013a), i.e., a kind of Estimation of Distribution Algorithms (EDAs) that make use of a purposely simple probabilistic model. Distinct from other types of EDAs, in a compact algorithm each variable of the problem is handled by a separate probability distribution. Moreover, compact algorithms rely on a very small number of solutions. This purposefully simple design is what allows for a lower memory footprint. However, as shown in (Iacca and Caraffini, 2020), this benefit comes with a cost. Firstly, compact algorithms tend to suffer from premature convergence, as they do not explicitly maintain diversity. This is because they do not keep an actual population (and all the solutions therein). Secondly, these algorithms show a poorer performance on non-separable problems, compared to population-based metaheuristics. This is due to the fact that they process each variable separately, by using (typically) a univariate Gaussian distribution for each variable. Despite these limitations, compact algorithms have been proven successful in a number of applications, such as convolutional neural networks (Singh and Paul, 2021) and large scale optimization (Ferigo and Iacca, 2020).

One possible way to overcome the shortcomings observed in compact algorithms consists in hybridizing them with other search techniques. However, in doing so, in principle one should try to preserve the original simplicity of compact algorithms. Previous works proved that the appropriate hybridization of lightweight metaheuristics can obtain promising results. For instance, in (Caraffini et al, 2013; Iacca et al, 2012c) the authors demonstrated that simple single-solution algorithms composed of two or three search single-solution strategies can outperform complex population-based algorithms.

Inspired by these successful instances of lightweight metaheuristics, in the present paper a similar hybridization principle is adopted. Our goal is to design a new memetic algorithm capable to improve upon the performance of compact algorithms. The main difference between our proposal and the previous literature is that here we hybridize, in a memetic fashion, a compact algorithm with a single-solution algorithm method, rather than multiple single-solution algorithms. The rationale behind this design is to combine the advantage of the compact logic with that of a single-solution algorithm.

The memetic algorithm proposed here uses the Non-Uniform Mutation (NUM), i.e., a mutation operator that searches the space uniformly at the early stages of the optimization process, and more locally at the later ones (Zhao et al, 2007). This dynamic behavior allows to improve the fine-tuning capabilities of the algorithm while also reducing the disadvantages of the canonical mutations used in compact optimization. In the original implementation of NUM, multiple (actually, all) variables are perturbed at the same time: hence, in what follows this variant will be referred to as MNUM (where "M" stands for "Multi"). On the other hand, the recent work (Khalfi et al, 2022) demonstrated that, by simplifying the NUM logic so to handle one variable at a time, it is possible to effectively deal with separable problems in both single-solution and compact optimization settings. This NUM variant was dubbed as SNUM (where "S" stands for "Single"). Following that work, the goal of the present study is to combine the strengths of SNUM and MNUM into a compact optimization memetic scheme. As will be shown later, the coexistence of both variants provides different but complementary roles. It is worth noticing that, while the NUM operator has already been used in the previous literature, it has never been studied and applied in a compact optimization scheme as in this work. In a nutshell, the main contributions of this work can be summarized as follows:

1. A novel memetic metaheuristic based on the NUM operator and influenced by two different logics (based on a probabilistic model and a single solution). The proposed algorithm is dubbed as "compact Single/Multi Non-Uniform Mutation algorithm" (in short, cSM).
2. An ablation study aimed at analyzing the effectiveness of each component of the algorithm independently, and illustrating the importance of the two different logics in the same underlying structure.
3. A comparison of the cSM performance with respect to 12 state-of-the-art compact and single-solution algorithms (including 4 memetic algorithms), on 4 benchmarks (BBOB, CEC-2014, CEC-2017 and CEC-2011) at different dimensionalities.
4. A comparison of the cSM performance with respect to 4 state-of-the-art population-based algorithms (namely, the winners of the CEC-2014, CEC-2017 and CEC-2011 competitions), aimed at unfolding the potential and limitations of cSM.

The rest of the paper is structured as follows. Section 2 provides an overview of the field of Memetic Computing, as well as the basics of compact optimization algorithms. Section 3 describes the working principles and algorithmic details of cSM. The experimental study and detailed analysis of the numerical results are then given in Section 4. Finally, Section 5 concludes this work and suggests some possible future research directions.

2 Related works

This section briefly summarizes the related works and background concepts in the field of Memetic Computing (Section 2.1) and compact optimization (Section 2.2).

2.1 Memetic Computing

While researchers often design new metaheuristics by proposing new search operators or strategies, another significant algorithmic design direction is to combine multiple existing algorithms/operators in the expectation that the resulting algorithmic framework will adapt to a broader variety of problems. Algorithms based on this view are often referred to as hybrid or *memetic* algorithms (MAs). In the very first implementations of MAs, these algorithms were often designed following a predetermined structure, i.e., a population-based metaheuristic—typically, a Genetic Algorithm (GA)—to ensure a proper global search capability, combined with one or more local search techniques activated throughout the evolutionary cycle. As such, these algorithms are characterized by explicit exploration and exploitation phases, whose balance is known to be a crucial element in numerical optimization.

Differently from canonical MAs, in the modern view of Memetic Computing (MC) the population-based part of the algorithm may be missing. Instead, the algorithm can be built on several consecutive local searches, or simple search operators that perturb a single solution. In this sense, the definition of MC is much broader than the traditional MA paradigm: in fact, a MC approach is viewed as a composition of multiple modules—referred to as *memes*—which interact and cooperate to perform efficient problem solving. These algorithms have been proven successful in a number of real-world problems, such as scheduling (Decerle et al, 2019) and logistics (Eremeev and Kovalenko, 2020).

From the perspective of lightweight algorithms, several hybrid algorithms have been reported in the literature, in which Simulated Annealing (SA) notably one of the most popular single-solution algorithms, is combined with either another single-solution algorithm or with a population-based algorithm. For example, the authors of (Lenin et al, 2016) have presented a hybrid approach for solving the optimal reactive power problem that combines SA with Tabu Search. Others combined SA with a population-based algorithm while maintaining the same search operators of the latter, i.e., by performing cooperation/competition among the solutions in the population to effectively explore the search space. For instance, the authors of (Adler, 1993) presented a hybrid method based on SA and a GA. The goal of this hybrid SA-GA algorithm is to enhance the efficiency of the GA selection and crossover operators by employing the Metropolis selection rule typically used in SA.

2.2 Compact optimization

As discussed in the introduction, compact metaheuristics are stochastic search algorithms belonging to the Estimation of Distribution Algorithms (EDAs) family.Compact optimization algorithms employ a probabilistic model, referred to as Probability Vector ($\boldsymbol{PV}$), whose structure and cardinality depend on the problem variables' type (e.g., real or binary values) and dimensionality. This probabilistic model allows for a low memory footprint: in fact, instead of using an entire population, compact algorithms use $\boldsymbol{PV}$ to sample at each iteration a small number of solutions, which are then used to update $\boldsymbol{PV}$.

For the case of binary-encoded compact GA (cGA) (Harik et al, 1999), $\boldsymbol{PV}$ is an D-dimensional vector represented by $\boldsymbol{p} = [p_1, p_2, \ldots, p_D]$ where each $p_i \in [0,1]$ $i = 1,2,\ldots,D$ indicates the probability that the i-th variable is set to 1, and D is the problem dimensionality. In the real-valued compact GA (rcGA) (Mininno et al, 2008), on the other hand, a truncated normalized Gaussian Probability Density Function (PDF) is used as $\boldsymbol{PV}$, which in this case is a $2 \times D$ matrix containing the mean vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_D]$, and the standard deviation vector $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_D]$, where each i-th element of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ refers to the i-th variable. Both cGA and rcGA depend on a hyperparameter, i.e., the virtual population size N_P.

The aforementioned cGA was the first compact metaheuristic to be devised, mainly to target binary optimization problems. In (Harik et al, 1999), cGA was reported to perform almost as well as a population-based GA. Since then, several new cGA variants have been proposed. Most of these attempts share the same algorithmic structure of the original cGA, but introduce new concepts/operators and combinations thereof, which eventually allow to outperform the original cGA.

Similarly, rcGA originated several research directions related to compact continuous optimization. Most of the research in this field focus on the development of new compact versions of population-based metaheuristics and/or try to enhance existing methods. Table 1 summarizes some of the main compact algorithms introduced so far in the literature. All works emphasize the fact that these algorithms are simple and easy to be implemented: something that has, not surprisingly, led to a variety of applications. At present, compact metaheuristics have found successful applications in a wide range of real-world scenarios, see for instance (Neri et al, 2011; Mininno et al, 2011; Iacca et al, 2012a; Tighzert et al, 2018; Yang et al, 2018). It should be noted, however, that while some of these scenarios are actually memory-limited, in other cases memory is not a real issue, yet compact algorithms found a possibility for application.

Despite the differences between the various existing compact metaheuristics, it has been shown that these algorithms all adhere to a somehow unified search logic (Neri et al, 2013a), which, analogously to single-solution and population-based metaheuristics, essentially attempts to keep a good balance between exploration and exploitation. From a design perspective, the search procedure of any compact algorithm is rather straightforward and comprises three main stages, working as follows (for illustration purposes, the vanilla version of rcGA is considered below):

Stage 1) A Gaussian Probability Distribution Function (PDF) (truncated inside the range $[-1,1]$) is considered for each i-th variable, with mean μ_i and standard deviation σ_i taken from $\boldsymbol{PV} = [\boldsymbol{\mu}; \boldsymbol{\sigma}]$. The PDF height is normalized such that its area is equal to 1. For each i-th variable, μ_i and σ_i are initially set to 0 and λ respectively, where λ is an arbitrarily big positive constant (typically $\lambda = 10$), in order to replicate a uniform distribution in $[-1,1]$. This setting allows to explore the search space at the beginning of the optimization process. Afterwards, a starting solution, $\boldsymbol{elite}$, is generated by sampling each i-th variable from its associated PDF.

Stage 2) The iterative process starts in this stage, and this is where the distinction among compact algorithms becomes apparent. In general, a candidate solution $\boldsymbol{x_{off}}$ is generated by sampling one or more solutions (typically, one to three) from the current $\boldsymbol{PV}$, depending on the specific compact algorithm logic. The operators of the latter (mutation, crossover, etc.) are then applied, and the fitnesses of $\boldsymbol{elite}$ and $\boldsymbol{x_{off}}$ are compared. The solution with the best fitness value among the two is set as $\boldsymbol{winner}$, and the one showing the worst fitness value is set as $\boldsymbol{loser}$. Finally, $\boldsymbol{elite}$ is substituted by $\boldsymbol{x_{off}}$, depending on the selected replacement criterion, that can be based on either a persistent or a non-persistent elitism scheme.

Stage 3) The result of this fitness comparison (i.e., between $\boldsymbol{winner}$ and $\boldsymbol{loser}$) is then used to update the values of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, in such a way to “push” the Gaussian PDF in the direction of the better solution, and “shrink” it around it.

The cyclic application of Stages 2 and 3 continues until a certain stop condition is met. Of note, the sampling and the update mechanisms are a common aspect for all compact algorithms, see (Mininno et al, 2008) for details.

Table 1 Some of the most popular real-valued compact algorithms (sorted by year).

Long name of the compact algorithm	**Short name**	**Year**	**Ref.(s)**
compact Genetic Algorithm	rcGA	2008	(Mininno et al, 2008)
compact Differential Evolution with binomial/exponential crossover	cDE, cDE-Exp	2011	(Mininno et al, 2011)
compact Bacterial Foraging Optimization	cBFO	2012	(Iacca et al, 2012b)
compact Particle Swarm Optimization	cPSO	2013	(Neri et al, 2013b)
compact Teaching-Learning-Based Optimization	cTLBO	2014	(Yang et al, 2018)
Enhanced compact Artificial Bee Colony	EcABC	2015	(Banitalebi et al, 2015)
compact Firefly Algorithm	cFA	2018	(Tighzert et al, 2018)

Among the various compact optimization algorithms introduced in the literature, cDE has played a major role in the field, not only because it has inspired many other works to apply the concept of compact optimization to other metaheuristics (as shown in Table 1), but also because it has given rise to many variants. A short description of some of them is given in Table 2.

Table 2 Some of the most important compact Differential Evolution variants.

Long name of the compact algorithm	**Short name**	**Short Description**	**Ref.(s)**
Memetic compact Differential Evolution	McDE	Employs a stochastic local search algorithm to iteratively modify the search direction.	(Neri and Mininno, 2010)
Disturbed Exploitation compact Differential Evolution	DEcDE	Combines an intensively exploitative/evolutionary search based on cDE with a shallow-depth exploitative local search.	(Neri et al, 2011)
compact Differential Evolution light without and with non-disruptive restart mechanism	cDE-light, RI-cDE-light	Proposes a lightweight mutation that allows sampling of a single solution, instead of multiple solutions as in the original cDE, and a light crossover that requires less random numbers. RI-cDE-light is a cDE-light with a Re-Sampled Inheritance (RI) restart mechanism.	(Iacca et al, 2012a), (Iacca and Caraffini, 2020)
compact Compound Sinusoidal Differential Evolution	CScDE	Utilizes two sinusoidal formulas to automatically adjust the crossover rate and the mutation scaling factor in cDE.	(Khalfi et al, 2021)

Some memetic compact algorithms have also been proposed. Considering binary optimization, an extended compact GA (EcGA) in conjunction with the Nelder-Mead algorithm was presented for cluster optimization, see (Sastry and Xiao, 2001). More recently, the authors of (Xue and Chen, 2019) presented a compact Evolutionary Algorithm coupled with Tabu Search to address the sensor ontology matching problem efficiently. As for continuous optimization, to the best of our knowledge the main memetic compact algorithms proposed in literature are the Memetic compact Differential Evolution (McDE) algorithm introduced in (Neri and Mininno, 2010), the Disturbed Exploitation compact Differential Evolution (DEcDE) algorithm, introduced in (Neri et al, 2011), and the compact SNUM-based algorithm (cSNUM), introduced in (Khalfi et al, 2022).

3 Proposed bi-module memetic approach

Before discussing the details of our proposed cSM algorithm, the main motivation behind its design is explained in Section 3.1, followed by a description of the NUM operator that serves as the central component of the algorithm (see Section 3.2). Then, Section 3.3 presents the overall algorithm. For the sake of clarity, the Appendix (Table 29) reports the list of symbols introduced hereinafter.

3.1 Motivation behind the proposed algorithm

The main motivation for proposing a compact memetic algorithm is that, as mentioned above, only a few attempts have been done in this direction, particularly for continuous optimization problems. In addition, compact algorithms have been usually used as simple stand-alone approaches; on the contrary, the present study attempts: 1) to combine a compact and a single-solution algorithm; and 2) to coordinate them in a memetic fashion, in order to obtain a high-end lightweight optimizer especially suitable for non-separable functions. Furthermore, as will be clarified below, the mix SNUM/cMNUM has been chosen (instead of a combination of only compact components, i.e., cSNUM/cMNUM), in order to reduce the overhead introduced by the sampling mechanism needed in the compact algorithm components. Moreover, the roles of SNUM and cMNUM are complementary: in fact, SNUM proves to work better than cSNUM on problems characterized by uncorrelated variables (i.e., separable problems), while cMNUM is meant to handle non-separable problems.

3.2 Non-Uniform Mutation (NUM)

To overcome the limitations of random mutations in Evolutionary Algorithms, the dynamic NUM operator has been extensively used in the literature, see for instance (Zhao et al, 2007; Xinchao, 2011). This operator is defined as follows. Given $\boldsymbol{x}^t = [x_1, \dots, x_k, \dots, x_D] \in R^D$, that is a real-coded representation of the current solution (t being the iteration counter, i.e., the current number of function evaluations), and x_k, that is a variable randomly selected from $\boldsymbol{x}^t$, the result of the mutation is $\boldsymbol{x}^{t+1} = [x_1, \dots, x'_k, \dots, x_D]$, where:

$$x'_k = \begin{cases} x_k + \Delta(t,\ U_k - x_k), & \text{if } \eta = 1 \\ x_k - \Delta(t,\ x_k - L_k), & \text{if } \eta = -1 \end{cases} \tag{1}$$

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 x_k, respectively. The function $\Delta(t, y)$ returns a value in the range $[0, y]$ such that $\Delta(t,\ U_k - x_k)$ approaches to zero as the iteration counter t increases. This function is defined as follows:

$$\Delta(t, y) = y \times \left(1 - \rho^{(1-t/T)^B}\right) \tag{2}$$

where ρ is a uniform random number generated in $[0, 1]$, T is the total number of function evaluations, and B is a parameter determining the degree of dependency on the iteration counter (non-uniformity).

As analyzed in (Zhao et al, 2007), this mutation is neither a Gaussian mutation, which searches only locally (around its mean value), nor a Cauchy mutation, which takes long steps in the search space. Figure 1 illustrates how the step size (i.e., the radius of the mutation neighborhood) changes during the search process when the NUM operator is applied. As can be seen, NUM explores the whole space uniformly, with long jumps, at the early stages of the optimization process, while the length of the jumps decreases at the later stages. As a consequence, the algorithm explores increasingly smaller regions of the search space.

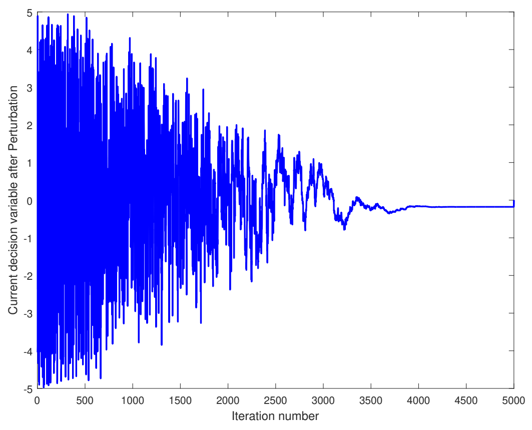


Fig. 1 Mutation step size over time on a 1-D sphere function with boundary $[-5, 5]$. The vertical axis shows the value of the current variable after the application of the NUM operator; the horizontal axis shows the iteration counter (number of function evaluations).

As anticipated in the introduction, when NUM is performed on a *single* variable at a time, the operator is referred to as Single Non-Uniform Mutation (SNUM) (Khalfi et al, 2022), as opposed to the original Non-Uniform Mutation where multiple (actually, all) variables of the current solution are perturbed at the same time, that is referred to as Multi Non-Uniform Mutation (MNUM).

3.3 Compact Single/Multi Non-Uniform Mutation algorithm (cSM)

The key idea of the proposed memetic algorithm is to use, inside the same structure, both SNUM and MNUM into two different modules (see sections 3.3.2 and 3.3.3), which are executed sequentially. The budget (i.e., number of function evaluations) for Module-1, $budget_1$, is set to 20% of the total budget T, while the remaining budget is used for Module-2 ($budget_2 = T - budget_1$). The first main feature of cSM is its basic algorithmic structure, composed of only two simple components, each one having a well-defined, specialized algorithmic role. The second primary advantage of cSM is its flexibility, i.e., its capability of dealing with both separable and non-separable problems, as will be shown in our experiments. In principle, the integration of the two modules in the same structure improves the overall global optimization capability. However, stagnation may still be possible with this algorithmic structure, as the algorithm exploits just a small number of solutions. The pseudo-code is shown in Algorithm 1. In what follows, the initialization phase of the algorithm and the two modules are discussed in detail.

3.3.1 Initialization

Similarly to what described in Section 2.2, the algorithm starts by initializing $\boldsymbol{PV}$, which is then used to randomly sample an initial “global best” solution, $\boldsymbol{elite}$, and a second solution, referred to as “local best”, $\boldsymbol{lbest}$, that is used only in Module-2. Then the budget for the two modules, respectively $budget_1$ and $budget_2$, is set. See lines 4-11 in the pseudo-code shown in Algorithm 1.

3.3.2 Module-1: SNUM

This module starts by copying $\boldsymbol{elite}$ into another solution $\boldsymbol{x_{off}}$. Then, the latter is perturbed using the SNUM operator, to generate a new solution, and the fitness of this new solution is evaluated right after the perturbation. The fact that only one variable is perturbed iteratively means that the problem is tackled *as if it were separable*. If the new solution outperforms the current $\boldsymbol{elite}$, the latter is replaced. This process is repeated (see lines 14-16 in the pseudo-code shown in Algorithm 1) until the first portion of the budget (i.e., $budget_1$) is exhausted. In this case, Module-2 is activated. It is important to notice here that Module-1 does not handle $\boldsymbol{PV}$, since it is a single-solution rather than a compact algorithm. However, for consistency with Module-2 (that instead is based on a compact logic), the solutions handled in Module-1 are kept in the same normalized range used for $\boldsymbol{PV}$, i.e., $[-1,1]$.

3.3.3 Module-2: cMNUM

When Module-2 is executed for the first time, $\boldsymbol{\mu}$ is initialized with $-\boldsymbol{elite}$ obtained from Module-1[1]. Hence, $\boldsymbol{elite}$ plays the role of information exchange from Module-1 to Module-2.

In essence, this second module allows for a good global search capability. Moreover, differently from Module-1, Module-2 includes the MNUM operator to handle non-separability and thus assist, in a memetic fashion, the first module. More precisely, a new solution, $\boldsymbol{x_{off}}$, is sampled from $\boldsymbol{PV}$. This solution then undergoes the application of the MNUM operator. This last operator, acting as a local search, constructs a neighborhood around $\boldsymbol{x_{off}}$, and replaces $\boldsymbol{x_{off}}$ with a new solution within that neighborhood. The local best, $\boldsymbol{lbest}$, is then replaced by this solution in case it is outperformed by it. At the end of this module, $\boldsymbol{PV}$ is updated, as well as $\boldsymbol{elite}$ (in case it is outperformed by $\boldsymbol{lbest}$). This procedure is repeated (see lines 24-37 in the pseudo-code shown in Algorithm 1) until the rest of allocated budget ($budget_2$) is exhausted.

4 Computational experiments

In order to evaluate the performance of the proposed cSM algorithm, four separate experiments on four well-known testbeds have been performed, in comparison with 12 well-established lightweight metaheuristics. Of note, the selected benchmarks comprise both synthetic test functions and real-world minimization problems. All the functions of the first three testbeds have shifted optima, while the optima of the problems in the last testbed are not known. The details of the four datasets are briefly summarized below.

- **BBOB benchmark** (Hansen et al, 2012). It consists of 24 test functions. The designers of the test-suite propose five dimensionalities to be treated (2, 3, 5, 10 and 20), and 40 as an optional choice. In the present study, two dimensionalities are considered: 20 and 40.
- **CEC-2014 benchmark** (Liang et al, 2013). It contains 30 test functions belonging to different classes, namely: unimodal (F1-F3), simple multimodal (F4-F16), six hybrid (F17-F22), and composition functions (F23-F30). With the exception of F8 and F10, all functions are non-separable. Four dimensionalities are considered in this study: 10, 30, 50 and 100.

[1] In preliminary experiments on the BBOB benchmark, not reported here for brevity, three cSM variants have been compared: one, where Module-2 was initialized with $-\boldsymbol{elite}$; one, where Module-2 was initialized with $\boldsymbol{elite}$; one, where the initialization in line 22 of Algorithm 1) was not present, thus $\boldsymbol{\mu}$ was kept at its initial values (zero). When looking at all the BBOB functions in 20 and 40 dimensions together, the numerical results obtained with the initialization at $-\boldsymbol{elite}$ were statistically better than those obtained with the other two variants. Hence, this variant is presented in the rest of this paper. Our intuition is that the initialization from an opposite $\boldsymbol{elite}$ allows Module-2 to explore a different part of the search space, thus counterbalancing the exploitation achieved in Module-1.

Algorithm 1 Pseudo-code of the proposed cSM algorithm.

```
1: Input: $\boldsymbol{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}], D$
2: Output: $\boldsymbol{elite}$
3: Parameters: $N_P$, $B$, $T$
4: for $i = 1:D$ do
5:     $\mu_i \leftarrow 0$
6:     $\sigma_i \leftarrow \lambda = 10$
7: end for
8: Generate $\boldsymbol{elite}$ by means of $\boldsymbol{PV}$        ▷ global best solution of the algorithm
9: Generate $\boldsymbol{lbest}$ by means of $\boldsymbol{PV}$        ▷ local best for Module-2
10: $budget_1 \leftarrow 20\%T$
11: $budget_2 \leftarrow T - budget_1$
12: while $t < T$ do
13:     if $t < budget_1$ then
14:         $\boldsymbol{x_{off}} \leftarrow \boldsymbol{elite}$
15:         $\boldsymbol{x_{off}} \leftarrow SNUM(\boldsymbol{x_{off}}, B, budget_1)$        ▷ NUM on a single variable, randomly selected (SNUM)
16:         $[\boldsymbol{winner}, \boldsymbol{loser}] \leftarrow\leftarrow compete(\boldsymbol{x_{off}}, \boldsymbol{elite})$
17:         if $\boldsymbol{x_{off}} = \boldsymbol{winner}$ then
18:             $\boldsymbol{elite} \leftarrow \boldsymbol{x_{off}}$
19:         end if
20:     else
21:         if $t = budget_1$ then
22:             $\boldsymbol{\mu} \leftarrow -\boldsymbol{elite}$        ▷ initial value of $\boldsymbol{\mu}$ for Module-2
23:         end if
24:         Generate $\boldsymbol{x_{off}}$ by means of $\boldsymbol{PV}$
25:         $\boldsymbol{x_{off}} \leftarrow MNUM(\boldsymbol{x_{off}}, B, budget_2)$        ▷ NUM on all variables (MNUM)
26:         $[\boldsymbol{winner}, \boldsymbol{loser}] \leftarrow compete(\boldsymbol{x_{off}}, \boldsymbol{lbest})$
27:         if $\boldsymbol{x_{off}} = \boldsymbol{winner}$ then
28:             $\boldsymbol{lbest} \leftarrow \boldsymbol{x_{off}}$
29:         end if
30:         for $i = 1:D$ do
31:             $\mu_i^{t+1} \leftarrow \mu_i^t + \frac{1}{N_P}(winner_i - loser_i)$
32:             $\sigma_i^{t+1} \leftarrow \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)}$
33:         end for
34:         $[\boldsymbol{winner}, \boldsymbol{loser}] \leftarrow compete(\boldsymbol{elite}, \boldsymbol{lbest})$
35:         if $\boldsymbol{lbest} = \boldsymbol{winner}$ then
36:             $\boldsymbol{elite} \leftarrow \boldsymbol{lbest}$
37:         end if
38:     end if
39: end while
```

- **CEC-2017 benchmark** (Awad et al, 2016). It contains 30 test functions belonging to different classes, namely: unimodal (f1-f3), simple multimodal (f4-f10), hybrid (f11-f20) and composition functions (f21-30). All functions are non-separable. Four dimensionalities are considered in this study: 10, 30, 50 and 100.
- **CEC-2011 benchmark** (Das and Suganthan, 2010). It contains 22 real-world optimization problems (both unconstrained and constrained). Constrained problems are handled by means of a static penalty function. The problems, taken from various industrial fields such as chemistry, telecommunications engineering, and scheduling, are characterized by different dimensionalities and present various challenges in terms of optimization. The main details can be found in (Das and Suganthan, 2010). In this study, the seven problems characterized by the highest dimensionalities are considered, i.e., T09, T11.1, T11.2, T11.7, T11.8, T11.9 and T11.10, whose dimensionality is equal to 126, 120, 216, 140, 96, 96 and 96 respectively.

Thus, on the whole, the algorithms have been tested on 91 (24+30+30+7) problems. For each test function of the BBOB benchmark, every algorithm has been executed over 15 runs, as recommended by the testbed designers. The COCO platform[2] has been used to ensure consistency of results. As for the CEC-2014 and CEC-2017 functions, each algorithm has been run 51 independent times on each problem, as indicated in the benchmark definitions. For the experiments on the CEC-2011 real-world problems, 25 independent runs have been carried out for each tested algorithm on each problem. For BBOB, CEC-2014 and CEC-2017, every single run has been executed for a total budget $T = 5000 \times D$ function evaluations, where D is the problem dimensionality. For the CEC-2011 problems, a total budget of $T = 150000$ function evaluations has been used, as recommended in the CEC competition. All the experiments have been conducted on a machine with an I7 2.4 GHz CPU and 16 GB of memory, running Matlab 2018 on Windows 10.

As a basis for comparison, the following algorithms have been considered, with their parameters set as recommended in the original papers (please note that for some algorithms' name the lowercase is intended, to stick to their original naming):

1. real-valued compact Genetic Algorithm (rcGA) (Mininno et al, 2008) with persistent elitism;
2. compact Differential Evolution (Mininno et al, 2011) (cDE-Exp) with persistent elitism, $rand/1$ mutation and exponential crossover;
3. Disturbed Exploitation compact Differential Evolution (Neri et al, 2011) (DEcDE) with $rand/1$ mutation, trigonometric mutation, exponential crossover and a perturbation mechanism of $\boldsymbol{PV}$;
4. Memetic compact Differential Evolution (Neri and Mininno, 2010) (McDE) with persistent elitism, DE/rand/1 mutation and binomial crossover;
5. Compound Sinusoidal compact differential Evolution (Khalfi et al, 2021) (CScDE) with persistent elitism, DE/rand/1 mutation, exponential crossover and adaptive control parameters;
6. compact Single Non-Uniform Mutation algorithm (Khalfi et al, 2022) (cSNUM) with persistent elitism and Single Non-Uniform Mutation;
7. compact Particle Swarm Optimization (Neri et al, 2013b) (cPSO);
8. compact Teaching-Learning Based optimization (Yang et al, 2018) (cTLBO) characterized by the absence of algorithm parameters that need to be tuned;
9. compact Firefly Algorithm (Tighzert et al, 2018) (cFA) with persistent elitism;
10. non-uniform Simulated Annealing (Xinchao, 2011) (nuSA);
11. Simplified Intelligence Single Particle optimization (Zhou et al, 2008) (ISPO);
12. Three Stage Optimal Memetic Exploration (Iacca et al, 2012c) (3SOME).

Thus, 13 algorithms, ours included, have been considered in the experimental analysis. The reason for selecting these algorithms is manifold. First and foremost, state-of-the-art compact evolutionary and swarm intelligence algorithms, i.e., methods that belong to the same family of cSM, have been considered. More specifically, rcGA (1) is the basic reference algorithm, while cDE-Exp, DEcDE and McDE (2, 3, and 4) are different (and more efficient) variants of the original cDE. Moreover, cDE-Exp and DEcDE were chosen because of the good performance observed in (Khalfi et al, 2021), while McDE was picked because it is a memetic compact algorithm. Furthermore, the comparisons include two other recently introduced compact algorithms, namely CScDE and cSNUM (5 and 6), as well as three recent compact swarm intelligence algorithms, namely cPSO, cTLBO and cFA (7, 8, and 9, respectively). Finally, the last three algorithms (10, 11, and 12) represent state-of-the-art single-solution algorithms. Among these, nuSA was chosen because it uses the same operator as our proposal, i.e., NUM (in particular, MNUM). However, it is important to highlight that, while nuSA accepts also a worsening of the current solution with a certain probability, in cSM the current best solution (i.e., $\boldsymbol{elite}$) is replaced only in case of improvement, see Algorithm 1. ISPO was selected because it has been shown to work well on separable problems. As for 3SOME, apart from its memetic design, it has been shown to perform well on a wide range of problems compared to several state-of-the-art more sophisticated metaheuristics.

As for the algorithms' parametrization, their shared parameters are set as follows. The virtual population size N_P is equal to 300 for all the compact algorithms (as suggested in the original papers): this value was indeed used in most compact algorithms, apart from cFA, for which $N_P = 10 \times D$ was

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

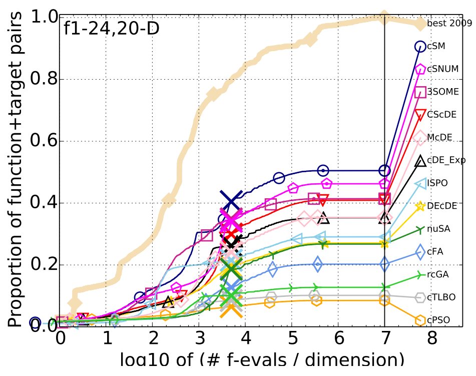
used instead, as recommended in (Tighzert et al, 2018). From a practical point of view, this parameter represents a step size ($1/N_P$) that controls the adjustment of the probabilistic model towards recently sampled good solutions (i.e., the winners), see lines 31-32 in Algorithm 1. Moreover, λ was set to 10 for all the considered compact algorithms, as recommended in (Mininno et al, 2008). As for cSM, B was set empirically to 6 (based on our parameter analysis reported in Section 4.1.2). In addition, the virtual population N_P was set to 1. A preliminary empirical analysis (not reported here for brevity) showed that setting N_P equal to 1 gives better results than $N_P = 300$, that is the default value used in most of the previous experiments with compact algorithms reported in the literature. The complete parameter settings for all the algorithms considered in our experimentation are reported in Table 3. From the table, it can be seen that cSM has less parameters to be tuned (in fact, just two: the non-uniform mutation factor B, and the budget for Module-1), compared to most of the other algorithms under study.

As a final remark, it should be mentioned that the performance of rcGA will be reported only for the BBOB benchmark, to have a more global view on how compact algorithms perform, while it will be omitted in the case of the CEC benchmarks. This is because in a preliminary analysis (not reported for brevity) this algorithm resulted clearly inferior to all the other algorithms included in our experimentation (in particular: 2, 3, 5, 6). It is also worth noticing that a toroidal handling of the search space bounds has been used in all the algorithms used in the study (see (Iacca et al, 2012c) for details), as done in most of the studies where these algorithms have been presented.

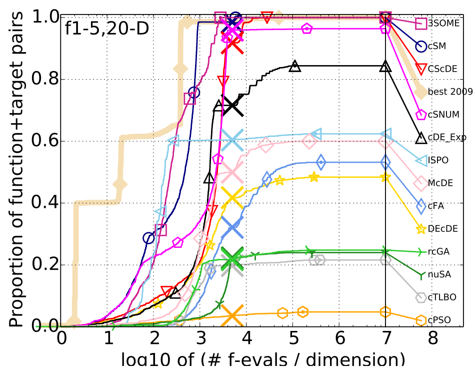
4.1 Results against lightweight metaheuristics on the BBOB benchmark

The first step of our comparative analysis consists in analyzing the performance of cSM against that of all the other lightweight algorithms under study on the BBOB benchmark. The main measure of performances used in this case is the Expected Running Time (ERT), that is automatically calculated by the COCO platform. This metric, used in Figures 2-3, depends on a given target function value, $f_t = f_{opt} + \Delta f$, where f_{opt} is the optimal function value (that is known for the BBOB functions) and Δf is a given precision to reach. It is computed as the number of function evaluations executed to reach f_t during each run, summed over all the runs, and divided by the number of runs that actually reached f_t (Hansen et al, 2012).

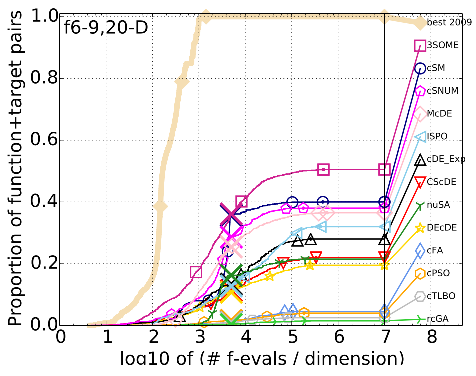
As can be seen from the Empirical Cumulative Distribution Function (ECDF) graphs in Figures 2-3, the proposed cSM obtained remarkably superior performances compared to all the other 12 algorithms. This is especially clear when looking at the overall set of functions (see Figure 2(a) and Figure 3(a), respectively for 20 and 40 dimensions). Moreover, cSM performed better in all the subgroups of functions, especially on moderate, ill-conditioned, multimodal and weakly-structured multimodal functions, where the other algorithms often failed. This better performance is likely due to the effect of the NUM operator (in both variants SNUM and MNUM), that allows cSM to better identify the directions of improvement over the fitness landscape. The only exception to this trend is the 3SOME algorithm, which showed almost similar performances on separable and moderate functions. When considering the whole set of functions, cSM could solve almost 40% of the targeted problems (f1-f24), even better than our recently proposed CScDE and cSNUM. On the other hand, most categories of functions resulted harder to solve for compact algorithms, compared to population-based metaheuristics. Another important remark is that the compact evolutionary algorithms used here for comparison (cDE-Exp, DEcDE, and CScDE) obtained better results than the swarm intelligence counterparts (cPSO, cFA, and cTLBO). Finally, when the dimensionality increases to 40, the performances of the cSM degrades only slightly, compared to the other algorithms that become instead much less efficient. Thus, the use of the NUM operator appears to preserve the scalability of the algorithm. This aspect will be further verified on higher dimensionalities on the CEC-2014 and CEC-2017 benchmarks.



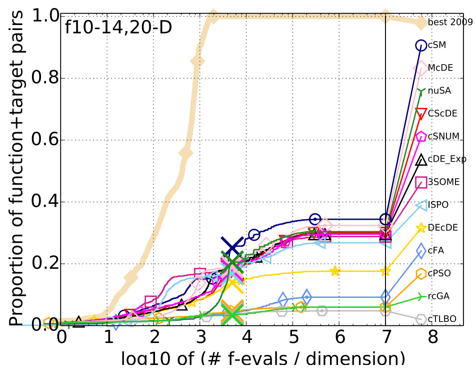
(a) All functions: f1-f24



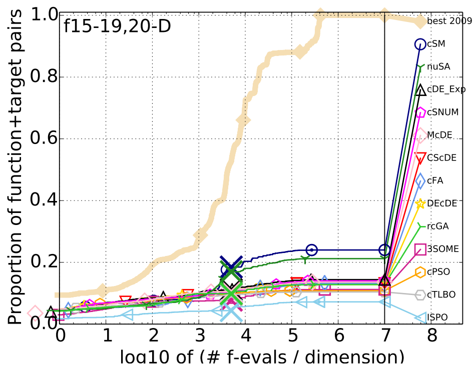
(b) Separable functions: f1-f5



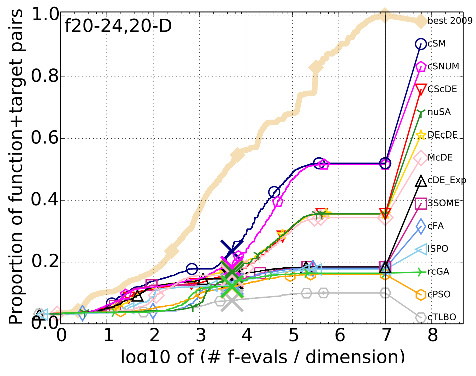
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

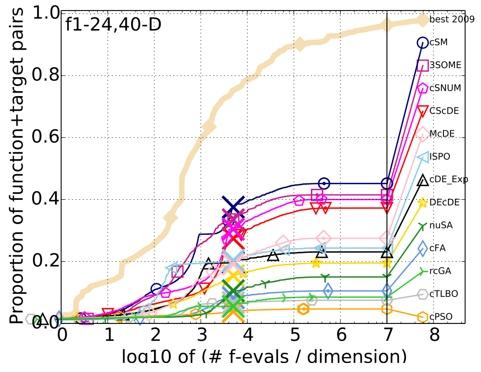


(e) Multimodal functions: f15-f19

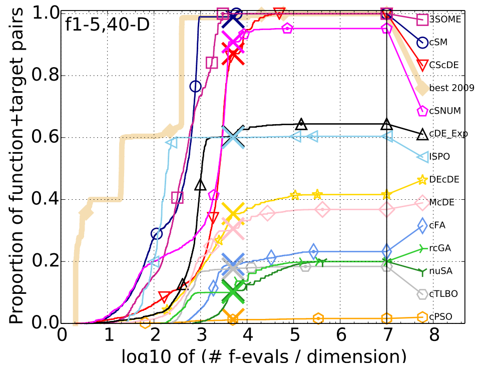


(f) Weakly-structured multimodal functions: f20-f24

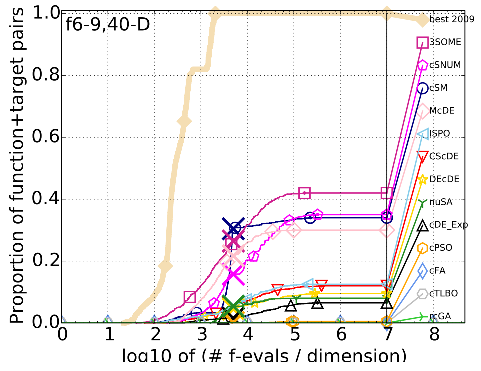
Fig. 2 Comparison of cSM vs lightweight algorithms in 20-D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimensionality (T/D) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the “best 2009” algorithm from the BBOB-2009 competition is also shown.



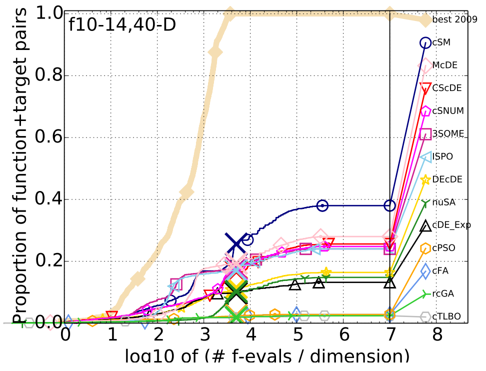
(a) All functions: f1-f24



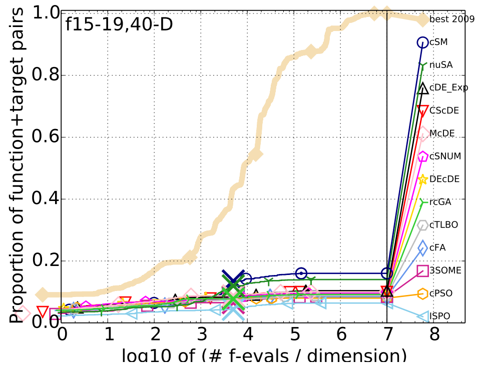
(b) Separable functions: f1-f5



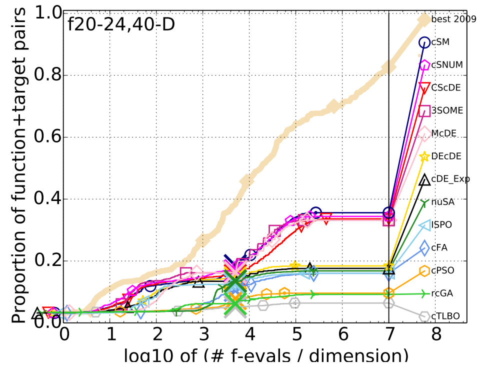
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14



(e) Multimodal functions: f15-f19



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

Fig. 3 Comparison of cSM vs lightweight algorithms in 40-D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimensionality (T/D) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from the BBOB-2009 competition is also shown.

Table 3 Parameter settings for the tested lightweight metaheuristics. Note that apart from nuSA, ISPO and 3SOME, all the other algorithms, including ours, also use an additional parameter N_P (i.e., the virtual population size).

Algorithm	**Parameter Name**	**Parameter Value**
rcGA	/	/
cDE-Exp	scale factor	$F = 0.5$
	crossover rate	$Cr = 1/\sqrt[\alpha_m D]{2}$
	proportion of genes undergoing exponential crossover	$\alpha_m = 0.25$
DEcDE	scale factor	$F = 0.5$
	crossover rate	$Cr = 1/\sqrt[\alpha_m D]{2}$
	trigonometric mutation probability	$Mt = 0.003$
	inheritance factor	$\alpha_m = 0.25$
	perturbation probability	$Mp = 0.001$
	maximum amplitude of perturbation	$\tau = 0.1$
McDE	scale factor	$F = 0.7$
	crossover rate	$Cr = 0.7$
	probability of local search activation	$p_{ls} 0.005$
	reduction factor	$\beta = 0.8$
	initial hypercube size	δ = 10% of the search space
CScDE	frequency of the sinusoidal formula	$freq = 1/D$
	number of waves being composed	$nWaves = log_2(D)$
cSNUM	non-uniform mutation factor	$B = 8$
cPSO	inertia weight	$W = -0.2$
	cognitive acceleration coefficient	$C_1 = -0.07$
	social acceleration coefficient	$C_2 = 3.74$
cTLBO	/	/
cFA	mutation coefficient	$\alpha = 0.25$
	damping ratio of mutation coefficient	$\alpha_{Dump} = 0.995$
	attractiveness of the firefly	$\beta_0 = 0.2$
	absorption coefficient	$\gamma = 1$
nuSA	mutation exponent	$B = 5$
	temperature reduction ratio	$\alpha = 0.9$
	temperature reduction period	$Lk = 3$
	number of initial solutions set to the initial temperature	$initialSolutions = 10$
ISPO	acceleration	$A = 1$
	acceleration power factor	$P = 10$
	learning coefficient	$B = 2$
	learning factor reduction ratio	$S = 4$
	minimum threshold on learning factor	$E = 1e-5$
	particle learning step	$PartLoop = 30$
3SOME	inheritance factor	$\alpha_e = 0.05$
	width of the hypercube for middle distance exploration	δ = 20% of the total search space width
	coefficient of generated solutions	$k = 4$
	initial exploratory radius for short distance exploration	ρ = 40% of the total search space width
cSM	non-uniform mutation factor	$B = 6$
	budget for Module-1	$budget_1 = 20\%T$

4.1.1 Effect of the two modules of the algorithm on the whole performance

To understand the rationale and the alleged coordination between the two modules, and study the effect of each module on the whole performance of cSM, three variants of the algorithm have been compared: 1) the complete variant that contains both modules, thus the above-described cSM; 2) a variant that uses only Module-1, named here SNUM; and 3) a variant that used only Module-2, labeled as cMNUM. The latter two variants are executed both with $B = 6$, for consistency with the parametrization of cSM. Figures 4 and 5 summarize the results of this comparison on the BBOB benchmark in 20 and 40 dimensions, respectively, again in the form of ECDF graphs.

As shown in the figures, the variant that executes both modules obtained better results. The figures that illustrate the performance of the algorithms for the case of functions f1-f5, in both dimensionalities (see Figure 4(b) and Figure 5(b)), show in particular that Module-1 allows the proposed cSM algorithm to effectively solve separable functions. On the other hand, the MNUM operator used in Module-2 permits to deal with functions with advanced complexity, especially ill-conditioned and multi-modal functions, see the figures related to functions f10-f14 (Figure 4(d) and Figure 5(d)) and f15-f19 (Figure 4(e) and Figure 5(e)), in both dimensionalities. These observations lead to the conclusion that both modules are complementary and that their hybridization allows to obtain good results on the whole set of problems.

4.1.2 Effect of the two parameters of the algorithm on the whole performance using BBOB benchmark

In this section, the effect of the two parameters of cSM, namely B and $budget_1$, is analyzed. To carry out this analysis, all functions (f1-f24) from the BBOB benchmark have been employed. To effectively highlight the influence of both parameters, all the conceivable combinations of $B \in \{1,2,6,10,20,50,100\}$ and $budget_1 \in \{20\%,50\%,80\%\}$ have been considered. Figure 6 shows the performance of cSM, on both $D = 20$ and 40 dimensions, with the different parameter settings considered. The N_P parameter in all settings is consistent with the value reported in Section 4 (i.e., $N_P = 1$). Overall, the figure indicates that different values of B and $budget_1$ can cause a significant change in the performance of cSM. In particular, the parameter setting used in the previous experiments on the BBOB benchmark (i.e., $budget_1 = 20\%$ and $B = 6$) obtains the best results and it is therefore the one adopted in the remaining experiments on the CEC benchmarks. One may also observe that B should not be too low (i.e., it should be (> 2)) nor too high (i.e., it should be (< 100)), since these values decrease the performance of cSM regardless the value of the budget used (at least for the tested values). The reason might be that a very small/large value of B prevents the search process from discovering the promising region that contains the solution exchanged from Module-1 to Module-2, therefore reducing the capability of the algorithm to successfully explore the whole search space.

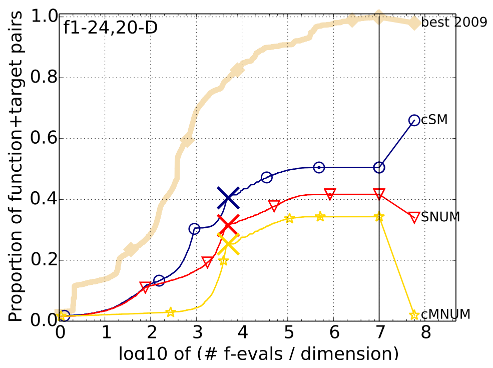
4.2 Results against lightweight metaheuristics on the CEC benchmarks

To evaluate the performance of the proposed cSM on the CEC benchmarks, two statistical tests, namely the Wilcoxon rank-sum test have been used to compare the algorithms and strengthen the interpretation of the numerical results.

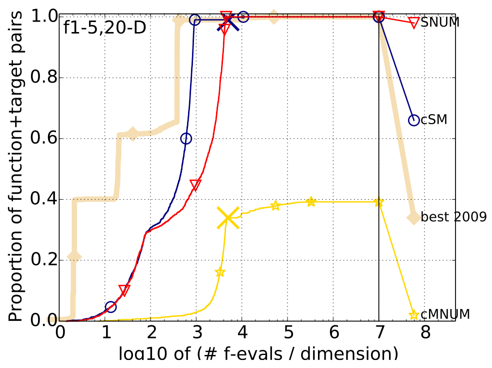
The Appendix reports the detailed results for the CEC-2014 and CEC-2017 benchmarks in terms of average fitness error and its standard deviation (across runs) for each algorithm, problem, and dimensionality. In all the tables reported in the Appendix, the boldface indicates the algorithm that obtained the minimum average error on each tested benchmark function (i.e., the minimum error between the best fitness obtained by the algorithm and the known optimum for that function). Furthermore, for each pairwise comparison between cSM (taken as reference) and all the other algorithms, the outcome of the Wilcoxon rank-sum test is shown, with a 5% significance level ($\alpha = 0.05$), see the columns labeled as "W". The symbol "+" indicates that the reference algorithm, cSM, statistically outperforms the competitor; "-" indicates that cSM is outperformed; and "=" means that the performances of the two algorithms are statistically equivalent.

4.2.1 Results on the CEC-2014 benchmark

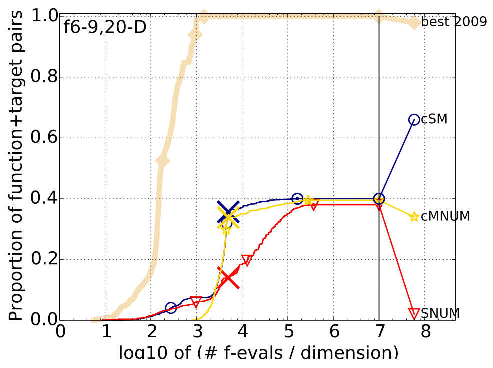
Tables 11-18, reported in the Appendix, show the results obtained when applying all the algorithms under comparison, ours included, on the CEC-2014 benchmark for 10, 30, 50 and 100 dimensions. For the sake of clarity, a compact representation of these results is reported in Table 4. Inspecting the numerical results, it can be seen that the proposed cSM algorithm is statistically superior to all the



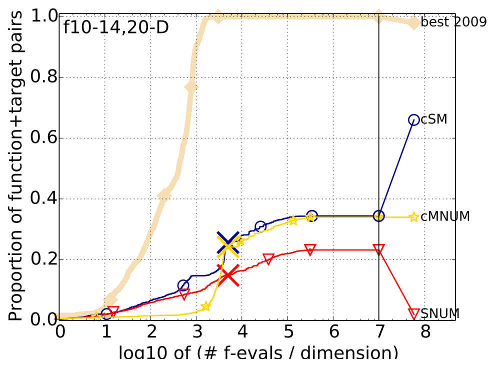
(a) All functions: f1-f24



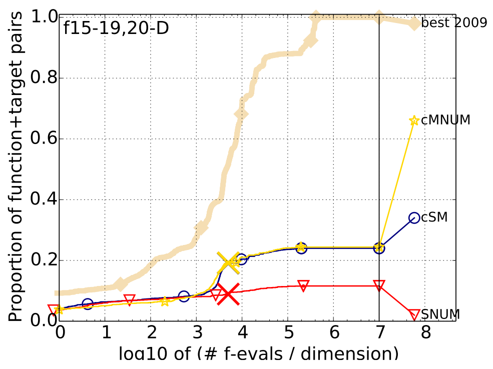
(b) Separable functions: f1-f5



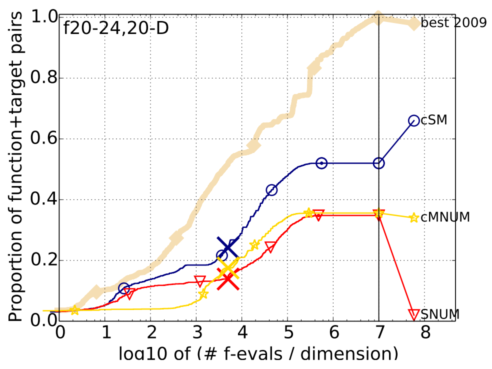
(c) Moderate functions: f6-f9



(d) Ill-conditioned functions: f10-f14

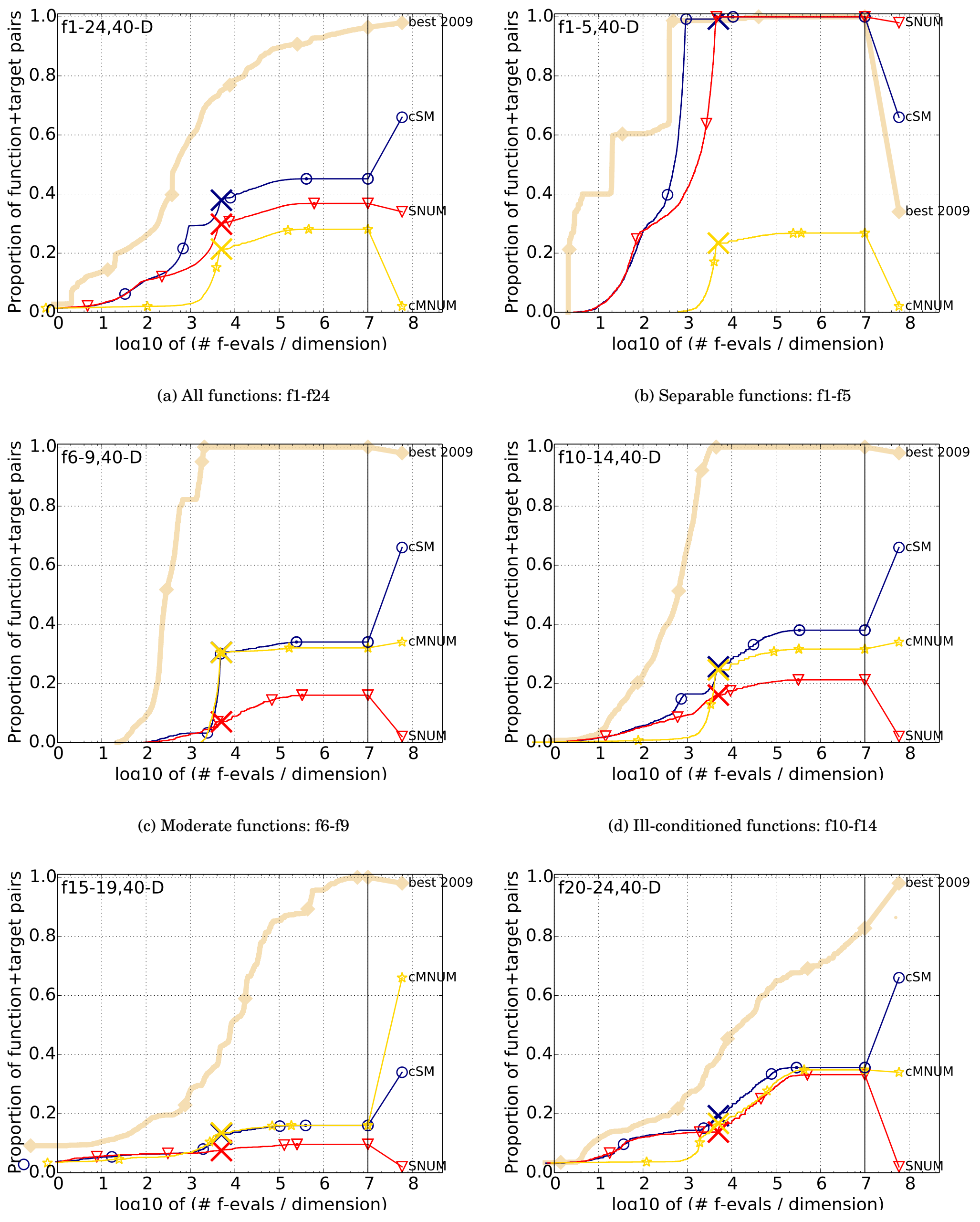


(e) Multimodal functions: f15-f19



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

Fig. 4 Effect of Module-1 (SNUM) and Module-2 (cMNUM) on the whole performance of cSM in 20-D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimensionality (T/D) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from the BBOB-2009 competition is also shown.



(a) All functions: f1-f24

(b) Separable functions: f1-f5

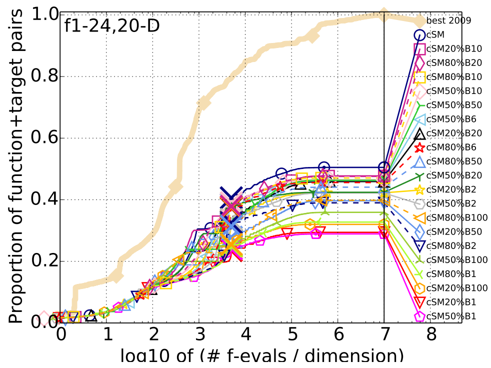
(c) Moderate functions: f6-f9

(d) Ill-conditioned functions: f10-f14

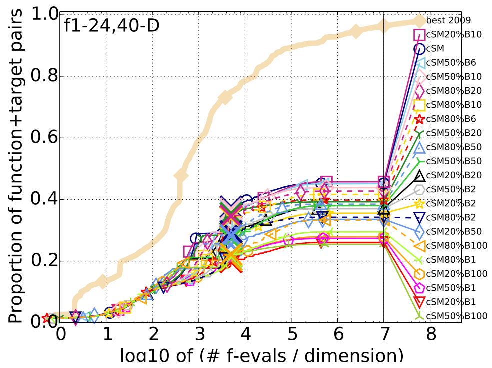
(e) Multimodal functions: f15-f19

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

Fig. 5 Effect of Module-1 (SNUM) and Module-2 (cMNUM) on the whole performance of cSM in 40-D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimensionality (T/D) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the “best 2009” algorithm from the BBOB-2009 competition is also shown.



(a) All functions: f1-f24 ($D = 20$)



(b) All functions: f1-f24 ($D = 40$)

Fig. 6 Effect of the parameters of the algorithm on the whole performance of cSM in 20-D and 40-D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimensionality (T/D) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, the "best 2009" algorithm from the BBOB-2009 competition is also shown. "cSM" indicates the default parametrization ($budget_1 = 20\%$ and $B = 6$).

other algorithms on the majority of the test functions in 10, 30, 50 and 100 dimensions. The superiority of cSM is substantial in most of the cases, even compared to the recent CScDE algorithm and the two other algorithms using the same NUM operator, i.e., cSNUM and nuSA. According to the results of the statistical test shown in Table 4, cSM appears significantly better, when considering the four dimensionalities together, than cDE-Exp, McDE, DEcDE, CScDE, cSNUM, cFA, cPSO, cTLBO, ISPO, nuSA and 3SOME in 91, 115, 90, 73, 62, 114, 117, 118, 95, 79 and 71 cases out of 120, respectively.

As seen from the Holm-Bonferroni procedure reported in Table 5, cSM (the best ranked one) performs better than all the other lightweight algorithms, for which the null hypothesis is rejected. It is also clear that cTLBO, McDE, cPSO, cFA and ISPO give the worst performances among all 13 algorithms.

Table 4 Number of statistically significant losses (-), draws (=), and wins (+) of cSM against cDE-Exp, DEcDE, McDE, CScDE, cSNUM, cFA, cPSO, cTLBO, ISPO, nuSA and 3SOME on the CEC-2014 benchmark (Liang et al, 2013) in 10, 30, 50 and 100 dimensions.

	cDE-Exp	**McDE**	**DEcDE**	**CScDE**	**cSNUM**	**cFA**	**cPSO**	**cTLBO**	**ISPO**	**nuSA**	**3SOME**
CEC-2014 (10-D)	6/11/13	1/4/25	5/7/18	5/6/19	4/10/16	2/2/26	0/3/27	0/2/28	0/2/28	4/14/12	4/8/18
CEC-2014 (30-D)	2/4/24	0/0/30	4/3/23	7/2/21	6/8/16	0/2/28	0/0/30	0/0/30	2/5/23	3/4/23	6/6/18
CEC-2014 (50-D)	0/5/25	0/0/30	3/4/23	7/6/17	10/3/17	0/0/30	0/0/30	0/0/30	3/3/24	4/3/23	7/5/18
CEC-2014 (100-D)	1/0/29	0/0/30	1/3/26	9/5/16	12/5/13	0/0/30	0/0/30	0/0/30	5/5/20	6/3/21	7/6/17
TOT (-/=/+)	9/20/91	1/4/115	13/17/90	28/19/73	32/26/62	2/4/114	0/3/117	0/2/118	10/15/95	17/24/79	24/25/71

4.2.2 Results on the CEC-2017 benchmark

Similarly to the experiments on CEC-2014, Tables 19-26 in the Appendix report the results obtained when applying all the algorithms under comparison, ours included, on the CEC-2017 benchmark for 10, 30, 50 and 100 dimensions. A concise summary of the results is reported in Table 6. The statistical results confirm the previously observed trend: on the vast majority of test functions, across all dimensionalities, cSM is statistically superior to all the other algorithms considered in the comparison. In particular, cSM is significantly better, when considering all dimensionalities, than cDE-Exp, McDE,

Table 5 Holm-Bonferroni procedure for the CEC-2014 benchmark (Liang et al, 2013) in 10, 30, 50 and 100 dimensions (reference algorithm: cSM, Rank = 1.05E+01).

j	**Optimizer**	**Rank**	z_j	p_j	σ/j	**Hypothesis**
1	CScDE	8.83E+00	-3.62E+00	1.49E-04	5.00E-02	Rejected
2	cSNUM	8.77E+00	-3.74E+00	9.14E-05	2.50E-02	Rejected
3	3SOME	8.19E+00	-4.98E+00	3.23E-07	1.67E-02	Rejected
4	nuSA	8.11E+00	-5.16E+00	1.26E-07	1.25E-02	Rejected
5	DEcDE	8.05E+00	-5.28E+00	6.41E-08	1.00E-02	Rejected
6	cDE-Exp	7.25E+00	-7.00E+00	1.28E-12	8.33E-03	Rejected
7	ISPO	4.65E+00	-1.26E+01	1.27E-36	7.14E-03	Rejected
8	cFA	4.59E+00	-1.27E+01	2.57E-37	6.25E-03	Rejected
9	cPSO	3.53E+00	-1.50E+01	3.53E-51	5.56E-03	Rejected
10	McDE	3.28E+00	-1.55E+01	9.34E-55	5.00E-03	Rejected
11	cTLBO	2.53E+00	-1.72E+01	3.09E-66	4.55E-03	Rejected

DEcDE, CScDE, cSNUM, cFA, cPSO, cTLBO, ISPO, nuSA and 3SOME in 91, 116, 83, 63, 64, 116, 116, 118, 95, 70 and 81 cases out of 120, respectively.

A further confirmation of this conclusion is obtained with the Holm-Bonferroni procedure. As seen from Table 7, cSM (the best ranked one) performs better than all the other lightweight algorithms, for which the null hypothesis was rejected. It is also obvious that cFA, ISPO, cPSO, McDE, and cTLBO show the poorest performances among the 13 algorithms, in the same way as they did on CEC-2014, although in a slightly different order. One also can notice that CScDE and cSNUM have the same position on both benchmarks, outperforming all the other algorithms except for the proposed cSM.

Table 6 Number of statistically significant losses (-), draws (=), and wins (+) of cSM against cDE-Exp, DEcDE, McDE, CScDE, cSNUM, cFA, cPSO, cTLBO, ISPO, nuSA and 3SOME on the CEC-2017 benchmark (Awad et al, 2016) in 10, 30, 50 and 100 dimensions.

	cDE-Exp	**McDE**	**DEcDE**	**CScDE**	**cSNUM**	**cFA**	**cPSO**	**cTLBO**	**ISPO**	**nuSA**	**3SOME**
CEC-2017 (10-D)	4/5/21	1/3/26	3/6/21	2/9/19	10/5/15	1/2/27	0/3/27	0/1/29	1/1/28	2/16/12	6/7/17
CEC-2017 (30-D)	5/6/19	0/0/30	4/9/17	6/7/17	7/7/16	0/1/29	1/0/29	0/1/29	4/4/22	4/11/15	7/3/20
CEC-2017 (50-D)	1/7/22	0/0/30	2/8/20	8/8/14	8/5/17	0/0/30	0/0/30	0/0/30	4/4/22	0/10/20	5/5/20
CEC-2017 (100-D)	0/1/29	0/0/30	2/3/25	8/9/13	8/6/16	0/0/30	0/0/30	0/0/30	5/2/23	2/5/23	3/3/24
TOT (-/=/+)	10/19/91	1/3/116	11/26/83	24/33/63	33/23/64	1/3/116	1/3/116	0/2/118	14/11/95	8/42/70	21/18/81

Table 7 Holm-Bonferroni procedure for the CEC-2017 benchmark (Awad et al, 2016) in 10, 30, 50 and 100 dimensions (reference algorithm: cSM, Rank = 1.02E+01).

j	**Optimizer**	**Rank**	z_j	p_j	σ/j	**Hypothesis**
1	CScDE	8.85E+00	-2.90E+00	1.86E-03	5.00E-02	Rejected
2	cSNUM	8.68E+00	-3.28E+00	5.26E-04	2.50E-02	Rejected
3	DEcDE	8.58E+00	-3.49E+00	2.41E-04	1.67E-02	Rejected
4	nuSA	8.24E+00	-4.21E+00	1.29E-05	1.25E-02	Rejected
5	cDE-Exp	7.33E+00	-6.16E+00	3.67E-10	1.00E-02	Rejected
6	3SOME	7.32E+00	-6.19E+00	2.93E-10	8.33E-03	Rejected
7	cFA	4.75E+00	-1.17E+01	5.77E-32	7.14E-03	Rejected
8	ISPO	4.29E+00	-1.27E+01	3.23E-37	6.25E-03	Rejected
9	cPSO	3.81E+00	-1.37E+01	3.29E-43	5.56E-03	Rejected
10	McDE	3.15E+00	-1.51E+01	4.04E-52	5.00E-03	Rejected
11	cTLBO	2.81E+00	-1.59E+01	4.37E-57	4.55E-03	Rejected

4.2.3 Results on the CEC-2011 benchmark

In order to further confirm the superiority of cSM and show its applicability to real-world problems, the proposed algorithm has been compared to the lightweight algorithms that showed the best performances in the previous analysis considering the CEC-2014 and CEC-2017 benchmarks together (namely, cDE-Exp, nuSA, 3SOME, CScDE and DEcDE), and included in this comparison also the top two algorithms that participated in the CEC-2011 competition, namely GA-MPC (Elsayed et al, 2011) and DE-ACr (Reynoso-Meza et al, 2011). This comparison has been done considering 7 of 22 real-world optimization problems of the CEC-2011 benchmarks (i.e., the ones with the highest dimensionality). Also in this case, the Wilcoxon rank-sum test and the Holm-Bonferroni procedure are used to evaluate the numerical results in statistical terms.

Tables 8 and 9 report the results of this comparison in terms of the average fitness ± standard deviation (along with the Wilcoxon rank-sum test) and the outcome of the Holm-Bonferroni procedure, respectively. From the tables, it can be observed that the proposed cSM performs statistically better/worse than cDE-Exp, nuSA, 3SOME, CScDE and DEcDE in 6/0, 7/0, 4/1, 3/1 and 3/1 out of 7 tested cases, respectively. As a note, since the raw data related to the top two algorithms from the CEC-2011 competition are not available, Table 8 shows only their mean values (as reported in their original papers), to illustrate how close the results obtained by algorithms that employ a limited number of solutions are to those obtained by algorithms using a population. Based on the mean fitness, DE-ACr appears to be the best-performing algorithm, although the official winner of the CEC-2011 competition was GAM-PC. It can also be seen that, compared to all the other considered algorithms, when cSM was surpassed, the mean fitness value achieved by cSM was not much higher than that obtained by the CEC-2011 winners, or the other algorithms. In addition, while all the other lightweight algorithms were not able to record any new best results in comparison with the winners, apart from 3SOME on problem 9, cSM performed better than all the other lightweight algorithms on problems 11, 18, 19 and 20. On the other hand, it is clear that cSM, as well as the other lightweight approaches, are not particularly tailored for the larger-scale problems included in the CEC-2011 benchmark.

Table 8 Average fitness ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against cDE-Exp, nuSA, 3SOME, CScDE and DEcDE on seven real-world optimization problems taken from the CEC-2011 benchmark (Das and Suganthan, 2010). The boldface indicates the lowest average fitness per each problem among the chosen lightweight algorithms. The underlined values indicate the lowest average fitness when GA-MPC (Elsayed et al, 2011) and DE-ACr (Reynoso-Meza et al, 2011) are included.

Function	**cSM**		**cDE-Exp**			**nuSA**			**3SOME**			**CScDE**			**DEcDE**			**GA-MPC**	**DE-ACr**
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Mean**
9	4.00E+03	3.74E+03	1.38E+04	1.01E+04	+	1.39E+06	5.19E+04	+	**8.05E+02**	3.34E+02	-	1.16E+03	7.57E+02	-	2.37E+03	1.83E+03	=	1.22E+03	3.35E+05
11	**5.24E+04**	5.68E+02	4.76E+05	1.15E+06	+	2.28E+05	3.32E+04	+	5.26E+04	4.61E+02	=	5.26E+04	5.86E+02	=	5.28E+04	6.09E+02	=	5.21E+04	4.63E+04
12	1.09E+06	2.30E+04	1.50E+06	1.47E+05	+	3.36E+06	9.85E+04	+	1.51E+07	1.11E+06	+	**1.08E+06**	1.57E+04	=	1.08E+06	1.30E+04	=	1.07E+06	1.05E+06
17	2.13E+06	1.98E+05	3.80E+08	4.80E+08	+	2.53E+06	3.93E+05	+	2.39E+06	4.11E+05	+	2.55E+06	1.15E+06	=	**2.06E+06**	3.05E+05	-	1.95E+06	1.84E+06
18	**1.04E+06**	2.61E+05	1.42E+06	6.51E+05	=	1.54E+07	3.37E+06	+	1.28E+06	6.33E+05	=	1.40E+06	5.35E+05	+	1.66E+06	8.61E+05	+	9.71E+05	9.24E+05
19	**1.52E+06**	3.04E+05	2.07E+06	6.91E+05	+	1.67E+07	2.99E+06	+	2.12E+06	8.36E+05	+	1.97E+06	6.80E+05	+	1.97E+06	5.84E+05	+	1.06E+06	9.30E+05
20	**1.01E+06**	1.79E+05	1.47E+06	4.76E+05	+	1.67E+07	3.48E+06	+	1.46E+06	7.14E+05	+	1.60E+06	6.02E+05	+	1.24E+06	4.29E+05	+	9.75E+05	9.24E+05
-/=/+				0/1/6			0/0/7			1/2/4			1/3/3			1/3/3			

Table 9 Holm-Bonferroni procedure on seven real-world optimization problems taken from the CEC-2011 benchmark (Das and Suganthan, 2010) (reference: cSM, Rank = 5.14E+00). Note that GA-MPC and DE-ACr are not included due to lack of raw data from their original papers.

j	**Optimizer**	**Rank**	z_j	p_j	σ/j	**Hypothesis**
1	DEcDE	4.14E+00	-1.00E+00	1.59E-01	5.00E-02	Not rejected
2	CScDE	4.00E+00	-1.14E+00	1.27E-01	2.50E-02	Not rejected
3	3SOME	3.86E+00	-1.29E+00	9.93E-02	1.67E-02	Not rejected
4	cDE-Exp	2.29E+00	-2.86E+00	2.14E-03	1.25E-02	Rejected
5	nuSA	1.57E+00	-3.57E+00	1.78E-04	1.00E-02	Rejected

4.3 Results against population-based metaheuristics on the CEC-2014 and CEC-2017 benchmarks

To draw conclusions concerning the performance of cSM with respect to populations-based algorithms, a comparison was made with respect to the winners of the CEC-2014 and CEC-2017 competitions, namely L-SHADE (Tanabe and Fukunaga, 2014) and JSO (Brest et al, 2017), which are both well-known to be robust and high-performing algorithms. For this comparison, the available raw data from the competitions have been used. Tables 27-28 report the detailed results in terms of average error ± standard deviation and pairwise Wilcoxon rank-sum test for cSM, L-SHADE and JSO on each benchmark problem in four dimensionalities (10, 30, 50 and 100). It is clear that the winners of the two competitions are the most effective algorithms on their respective benchmarks, which is expected given that these two algorithms are substantially more sophisticated than cSM. However, it is worth noticing that, while in low dimensionalities (i.e., 10 or 30 dimensions) cSM performs better than L-SHADE and JSO only in some isolated cases, its performance surprisingly improves for mid-scale dimensionalities (i.e., 50 to 100 dimensions).

4.4 Algorithmic complexity

The last step of our analysis consists in comparing the lightweight algorithms under consideration based on their execution times (see Section 4.4.1) and memory consumption (see Section 4.4.2).

4.4.1 Time complexity

Figure 7 shows the average (over 30 runs) computational time required to execute each algorithm under study with 200,000 evaluations of the f18 function from CEC-2014 in 10, 30, 50 and 100 dimensions. The results reveal that the algorithms have different computational overheads. Specifically, cSM and nuSA appear to have a higher execution time with respect to the other algorithms, while rcGA has the lowest overhead, followed by cSNUM. In addition, the compact swarm intelligence methods appear computationally cheaper than the cDE variants, but the former class of algorithms (cFA, cPSO and cTLBO) have been proven to be less effective. Although cSM, cSNUM and nuSA use the same operator, the fact that the NUM operator is applied to perturb all variables simultaneously increases the time complexity in cSM and nuSA. It is also important to consider that, despite the increase in the computational overhead of cSM with respect to other algorithms, its results are significantly better, because of the capability of the NUM operator to handle the non-separability of the objective function. This latter aspect is in fact not explicitly addressed in most of the other compared algorithms, which justifies their poorer performance.

4.4.2 Spatial complexity

Finally, the memory consumption of each algorithm considered in our experimentation has been examined. Besides the 13 lightweight algorithms (including ours), this analysis also includes the two winners of the CEC-2014 and CEC-2017 competitions. This comparison was done on the basis of the (approximate) number of D-dimensional vectors, hereinafter referred to as "slots", that each algorithm keeps in memory to conduct its operations.

The values shown in Table 10 indicate that all the lightweight algorithms use, as expected, much less memory than population-based algorithms. In fact, the number of slots being used in the case of lightweight algorithms is always fixed, regardless of the size of the problem. For instance, cSM always uses 5 D-dimensional slots (regardless the specific value of D), i.e., two slots for $\boldsymbol{PV}$, one slot for $\boldsymbol{elite}$, one slot for $\boldsymbol{lbest}$, and one slot for $\boldsymbol{x_{off}}$. Similar considerations can be made for all the other compact and single-solution metaheuristics considered in our experiments. On the other hand, for the

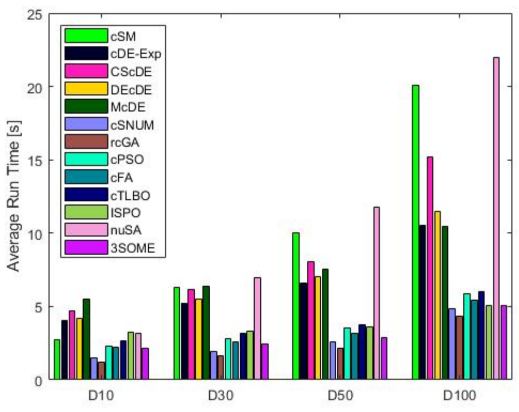


Fig. 7 Average execution time [s] for the lightweight algorithms under investigation, over increasing dimensionality values.

population-based CEC winners this value varies according to the dimensionality[3]. As discussed earlier, this difference in memory consumption is a major advantage of lightweight algorithms, including our proposed cSM, over population-based algorithms.

Table 10 Memory complexity in terms of the approximate number of memory slots (i.e., D-dimensional vectors) of the algorithms analyzed in the experimentation. Please note that, while for compact single-solution algorithms the memory footprint (in terms of no. of slots) does not depend on the D, the no. of slots for the internal memory structures used in L-SHADE and JSO explicitly depends on D.

Algorithm class	**Algorithm**	**Number of memory slots (≈)**					
		$D = 10$	$D = 20$	$D = 30$	$D = 40$	$D = 50$	$D = 100$
Compact optimization metaheuristics	cSM	5					
	CScDE	4					
	cSNUM	4					
	DEcDE	4					
	cDE-Exp	4					
	McDE	4					
	cFA	4					
	rcGA	4					
	cPSO	5					
	cTLBO	5					
Single-solution metaheuristics	nuSA	2					
	ISPO	2					
	3SOME	3					
Population-based metaheuristics	L-SHADE[a]	656	1306	1956	2606	3256	6506
	JSO[b]	369	675	937	1171	1389	2307

[a] $PopSize + ArchiveSize + HistoricalMemorySize = round(D \times rinit) + round(N^{init} \times r * arc) + 6 = 18 \times D + 47 \times D + 6$

[b] $2 \times PopSize + HistoricalMemory = 2 \times round\left(25 \times ln(D) \times \sqrt{D}\right) + 5$

[3] In particular, concerning L-SHADE (please see the original paper available at `http://metahack.org/CEC2014-Tanabe-Fukunaga.pdf`, Sec. III.A and Table 2), it results that the size (in terms of number of D-dimensional vectors) of the population and archive explicitly depends on D. Likewise, for JSO both the original paper and its publicly available source code (please see `https://github.com/justinjk007/JSO/blob/master/include/JSO.hpp`, line 32) indicate that the population size explicitly depends on D.

5 Conclusions and future works

In this work, a novel memetic compact algorithm has been presented. The proposed algorithm, dubbed cSM, is based on two forms of non-uniform mutation (NUM), acting either on a single variable (SNUM) or on multiple variables (MNUM); the former being encapsulated in a single-solution module, the latter in a compact optimization module.

Research contributions: The modular design principle adopted in this work provides a simple yet flexible framework that encapsulates two complementary search mechanisms. In our experimentation, cSM has displayed promising results, outperforming all competing lightweights algorithms especially when dealing with non-separable functions. Therefore, bringing compact algorithms and single-solution algorithms together by integrating them into a common framework can provide promising results. Another important aspect of cSM is the possibility to easily extend it. This feature, along with its good performances, makes cSM suitable for cheap and portable hardware. Therefore, cSM can have a significant impact in industrial environments characterized by severe computational limitations. Moreover, the algorithm contains just two modules, both with low memory complexity, and has a small number of parameters: B, $budget_1$, and N_P, the latter being needed in every compact algorithm. Having an algorithm with few parameters makes it easily applicable, because little or no effort to adjust its parameter setting for a specific problem is needed. Also these peculiarities make this algorithm advantageous.

Research limitations: When compared against state-of-the-art population algorithms, cSM could show respectable performance and was able to outperform, in some specific cases, even the winners of the CEC competitions. However, the substantial performance difference is still in favor of the population-based approaches, especially when all problems in all dimensionalities are considered. While NUM overcomes some of the limitations of compact algorithms, the absence of a population unquestionably impedes the attainment of better performances. Nevertheless, the promising results obtained by cSM will hopefully help those interested in achieving further advances in lightweight optimization.

Future works: As future research work, it might be useful to introduce a mechanism to make the budget allocation of cSM adaptive, instead of splitting it a priori between the two modules. In addition, it could be possible to replace each part of the algorithm with other existing operators/algorithms, in order to improve the results. Moreover, further performance improvements could be achieved by modifying some of the cSM inner elements, e.g. by including the same worsening acceptance criterion used in SA and nuSA. Another possibility would be to keep the basic cSM framework while trying to generalize it by adding more components (each one dealing with a specific category of problems), and finding an appropriate mechanism capable of deciding which module is particularly suitable for each specific problem at hand. Finally, designing parallel lightweight algorithms could be another important subject to research.

Statements and declarations

Author contributions: The authors' contribution in the different parts of the paper is as follows:
Souheila Khalfi: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Visualization; Resources; Data Curation; Writing - Original Draft; Writing - Review & Editing.
Giovanni Iacca: Conceptualization; Methodology; Writing - Review & Editing.
Amer Draa: Conceptualization; Methodology; Writing - Review & Editing.

Competing interests: The authors did not receive support from any organization for this work. The authors declare that they have no conflict of interest.
Data availability: The datasets generated and/or analyzed in the current study are available on reasonable request.

References

Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-Qaness MA, Gandomi AH (2021) Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput Ind Eng 157:107250

Adler D (1993) Genetic algorithms and simulated annealing: A marriage proposal. In: IEEE International Conference on Neural Networks, IEEE, pp 1104–1109

Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Method Appl M 391:114570

Awad H N, Ali Z M, Qu Y B, Liang J J, Suganthan N P (2016) 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

Banitalebi A, Aziz MIA, Bahar A, Aziz ZA (2015) Enhanced compact artificial bee colony. Inform Sciences 298:491–511

Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inform Sciences 237:82–117

Brest J, Maučec MS, Bošković B (2017) Single objective real-parameter optimization: Algorithm jso. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1311–1318

Caraffini F, Neri F, Passow BN, Iacca G (2013) Re-sampled inheritance search: high performance despite the simplicity. Soft Comput 17(12):2235–2256

Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Tech. rep., Jadavpur University, Nanyang Technological University, Kolkata

Decerle J, Grunder O, El Hassani AH, Barakat O (2019) A hybrid memetic-ant colony optimization algorithm for the home health care problem with time window, synchronization and working time balancing. Swarm Evol Comput 46:171–183

Elsayed SM, Sarker RA, Essam DL (2011) GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1034–1040

Eremeev AV, Kovalenko YV (2020) A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem. Memet Comput 12(1):23–36

Ferigo A, Iacca G (2020) A gpu-enabled compact genetic algorithm for very large-scale optimization problems. Mathematics 8(5):758

Hansen N, Auger A, Finck S, Ros R (2012) 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

Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. IEEE Trans Evol Comput 3(4):287–297

Iacca G, Caraffini F (2020) Re-sampled inheritance compact optimization. Knowl-based Syst 208:106416

Iacca G, Caraffini F, Neri F (2012a) Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead. J Comput Sci Technol 27(5):1056–1076

Iacca G, Neri F, Mininno E (2012b) Compact bacterial foraging optimization. In: Swarm and Evolutionary Computation, Springer, pp 84–92

Iacca G, Neri F, Mininno E, Ong YS, Lim MH (2012c) Ockham's razor in memetic computing: three stage optimal memetic exploration. Inform Sciences 188:17–43

Khalfi S, Draa A, Iacca G (2021) A compact compound sinusoidal differential evolution algorithm for solving optimisation problems in memory-constrained environments. Expert Syst Appl 186:115705

Khalfi S, , Iacca G, Draa A (2022) On the use of single non-uniform mutation in lightweight metaheuristics. Soft Comput 26:2259–2275

Lenin K, Reddy BR, Suryakalavathi M (2016) Hybrid tabu search-simulated annealing method to solve optimal reactive power problem. Int J Elec Power 82:87–91

Liang JJ, Qu BY, Suganthan PN (2013) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Tech. Rep. 635, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore

Mininno E, Cupertino F, Naso D (2008) Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Trans Evol Comput 12(2):203–219

Mininno E, Neri F, Cupertino F, Naso D (2011) Compact differential evolution. IEEE Trans Evol Comput 15(1):32–54

Neri F, Mininno E (2010) Memetic compact differential evolution for cartesian robot control. IEEE Comput Intell Mag 5(2):54–65

Neri F, Iacca G, Mininno E (2011) Disturbed exploitation compact differential evolution for limited memory optimization problems. Inform Sciences 181(12):2469–2487

Neri F, Iacca G, Mininno E (2013a) Compact optimization. In: Handbook of optimization, Springer, pp 337–364

Neri F, Mininno E, Iacca G (2013b) Compact particle swarm optimization. Inform Sciences 239:96–121

Oyelade ON, Ezugwu AES, Mohamed TI, Abualigah L (2022) Ebola optimization search algorithm: a new nature-inspired metaheuristic optimization algorithm. IEEE Access 10:16150–16177

Reynoso-Meza G, Sanchis J, Blasco X, Herrero JM (2011) Hybrid de algorithm with adaptive crossover operator for solving real-world numerical optimization problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1551–1556

Sastry K, Xiao G (2001) Cluster optimization using extended compact genetic algorithm. Tech. Rep. 2001016, IlliGAL

Singh L, Paul S (2021) Hybrid evolutionary network architecture search (hyenas) for convolution class of deep neural networks with applications. Expert Syst p e12690

Tanabe R, Fukunaga AS (2014) Improving the search performance of shade using linear population size reduction. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1658–1665

Tighzert L, Fonlupt C, Mendil B (2018) A set of new compact firefly algorithms. Swarm Evol Comput 40:92–115

Xinchao Z (2011) Simulated annealing algorithm with adaptive neighborhood. Appl Soft Comput 11(2):1827–1836

Xue X, Chen J (2019) Using compact evolutionary tabu search algorithm for matching sensor ontologies. Swarm Evol Comput 48:25–30

Yang Z, Li K, Guo Y, Ma H, Zheng M (2018) Compact real-valued teaching-learning based optimization with the applications to neural network training. Knowl-based Syst 159:51–62

Zhao X, Gao XS, Hu ZC (2007) Evolutionary programming based on non-uniform mutation. Appl Math Comput 192(1):1–11

Zhou J, Ji Z, Shen L (2008) Simplified intelligence single particle optimization based neural network for digit recognition. In: Chinese Conference on Pattern Recognition, IEEE, pp 1–5

A Detailed results on the CEC-2014 and CEC-2017 benchmarks

Table 11 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2014 in 10 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	1.01E+05	6.17E+04	1.22E+05	1.36E+05	=	5.32E+05	1.00E+06	=	7.64E+05	1.35E+06	+	1.38E+05	1.20E+05	=	**7.05E+04**	6.42E+04	-	7.94E+05	1.46E+06	+	3.24E+06	2.19E+06	+	5.92E+06	4.38E+06	+
f2	**2.47E+03**	2.66E+03	3.46E+03	3.44E+03	=	1.78E+07	1.10E+08	+	6.58E+03	4.46E+03	+	3.97E+03	3.51E+03	+	5.54E+03	4.00E+03	+	5.94E+03	3.55E+03	+	4.82E+07	2.13E+07	+	7.21E+08	4.34E+08	+
f3	**1.18E+03**	9.55E+02	6.84E+03	6.48E+03	+	6.86E+03	5.58E+03	+	6.73E+03	5.97E+03	+	6.85E+03	6.41E+03	+	6.32E+03	6.56E+03	+	1.28E+04	1.02E+04	+	6.31E+03	3.21E+03	+	1.14E+04	5.94E+03	+
f4	**9.25E+00**	1.51E+01	1.71E+01	1.95E+01	=	4.57E+01	2.37E+01	+	2.04E+01	1.90E+01	+	1.73E+01	1.81E+01	+	1.84E+01	1.69E+01	+	3.56E+01	2.16E+01	+	4.24E+01	1.78E+01	+	4.13E+01	1.21E+01	+
f5	2.00E+01	7.13E-05	**1.96E+01**	2.80E+00	-	2.01E+01	7.12E-02	+	2.00E+01	4.52E-03	+	2.00E+01	1.46E-03	+	2.00E+01	1.42E-04	-	2.00E+01	5.16E-02	-	1.99E+01	1.93E+00	+	2.04E+01	6.08E-01	+
f6	**1.83E+00**	1.34E+00	3.37E+00	1.25E+00	+	5.02E+00	1.38E+00	+	2.92E+00	1.16E+00	+	3.26E+00	1.28E+00	+	3.27E+00	1.75E+00	+	6.35E+00	1.48E+00	+	4.43E+00	1.41E+00	+	7.40E+00	1.21E+00	+
f7	**1.50E-01**	7.14E-02	1.90E-01	1.13E-01	=	5.42E+00	4.40E+00	+	2.00E-01	9.43E-02	+	1.69E-01	1.02E-01	=	1.70E-01	9.14E-02	=	3.96E+00	4.29E+00	+	2.14E+00	5.55E-01	+	6.87E+00	2.07E+00	+
f8	2.12E-13	8.51E-14	**6.46E-14**	5.69E-14	-	1.52E+01	5.03E+00	+	1.03E-03	9.36E-04	+	2.60E-08	1.86E-07	+	1.72E-13	9.48E-14	-	2.20E+01	9.33E+00	+	1.99E+01	8.44E+00	+	5.09E+01	8.98E+00	+
f9	1.28E+01	5.30E+00	1.39E+01	4.34E+00	=	2.85E+01	8.84E+00	+	1.32E+01	5.11E+00	=	1.34E+01	5.38E+00	=	**1.26E+01**	5.23E+00	=	2.85E+01	1.07E+01	+	3.74E+01	7.74E+00	+	5.68E+01	8.94E+00	+
f10	1.43E+00	2.08E+00	2.74E-01	4.75E-01	-	2.71E+02	1.56E+02	+	**1.57E-01**	4.79E-01	-	3.10E-01	6.96E-01	-	4.05E+00	1.68E+01	=	6.36E+02	3.68E+02	+	9.01E+02	1.87E+02	+	1.12E+03	2.66E+02	+
f11	5.42E+02	2.00E+02	5.25E+02	2.55E+02	=	7.27E+02	2.62E+02	+	5.11E+02	2.31E+02	=	5.00E+02	2.34E+02	=	**4.87E+02**	1.97E+02	=	8.81E+02	2.83E+02	+	1.23E+03	2.23E+02	+	1.43E+03	2.35E+02	+
f12	7.39E-02	6.68E-02	1.08E-01	7.18E-02	+	2.74E-01	1.21E-01	+	1.42E-01	5.99E-02	+	1.36E-01	8.53E-02	+	**6.80E-02**	5.04E-02	=	5.26E-01	3.25E-01	+	1.36E+00	1.83E-01	+	1.49E+00	3.15E-01	+
f13	**1.85E-01**	8.27E-02	3.04E-01	1.35E-01	+	4.59E-01	1.53E-01	+	3.41E-01	1.10E-01	+	3.77E-01	1.49E-01	+	3.54E-01	1.49E-01	+	4.24E-01	1.70E-01	+	5.17E-01	1.05E-01	+	7.62E-01	1.91E-01	+
f14	**1.92E-01**	7.61E-02	3.85E-01	2.35E-01	+	6.81E-01	5.17E-01	+	2.96E-01	1.79E-01	+	4.14E-01	2.78E-01	+	3.48E-01	2.10E-01	+	5.83E-01	3.93E-01	+	4.72E-01	2.11E-01	+	8.53E-01	4.03E-01	+
f15	1.12E+00	4.90E-01	1.76E+00	1.52E+00	+	1.37E+01	1.64E+01	+	1.56E+00	8.03E-01	+	1.75E+00	1.34E+00	+	**1.07E+00**	6.12E-01	=	6.87E+01	1.25E+02	+	5.43E+00	8.28E-01	+	3.89E+01	5.55E+01	+
f16	2.63E+00	3.80E-01	2.60E+00	3.52E-01	=	3.16E+00	3.92E-01	+	**2.44E+00**	4.20E-01	-	2.54E+00	4.74E-01	=	2.49E+00	4.35E-01	=	3.37E+00	3.24E-01	+	3.18E+00	2.38E-01	+	3.50E+00	2.43E-01	+
f17	**6.32E+03**	5.19E+03	9.14E+04	1.44E+05	+	4.38E+04	5.79E+04	+	3.08E+04	5.62E+04	+	1.14E+05	1.72E+05	+	1.13E+05	2.10E+05	+	5.47E+04	1.01E+05	+	7.41E+03	5.33E+03	=	4.09E+04	3.50E+04	+
f18	**5.33E+03**	8.52E+03	8.36E+03	9.78E+03	+	1.00E+04	1.04E+04	+	9.04E+03	1.11E+04	=	1.00E+04	1.03E+04	+	1.19E+04	1.23E+04	+	9.90E+03	1.22E+04	=	8.68E+03	8.95E+03	+	3.99E+04	3.64E+04	+
f19	1.48E+00	6.84E-01	1.44E+00	1.12E+00	=	2.67E+00	1.02E+00	+	1.37E+00	9.07E-01	=	1.29E+00	9.69E-01	-	**1.14E+00**	7.69E-01	-	4.77E+00	1.60E+00	+	3.75E+00	6.77E-01	+	5.10E+00	9.74E-01	+
f20	**1.18E+03**	3.31E+03	6.78E+03	9.20E+03	+	6.57E+03	8.70E+03	+	6.28E+03	9.15E+03	+	8.50E+03	1.07E+04	+	9.29E+03	1.06E+04	+	1.01E+04	1.19E+04	+	1.89E+03	2.60E+03	+	5.54E+03	5.25E+03	+
f21	**2.20E+03**	2.03E+03	7.74E+03	9.80E+03	=	6.16E+03	1.16E+04	=	5.51E+03	8.30E+03	=	1.02E+04	1.30E+04	+	1.05E+04	1.26E+04	+	7.46E+03	9.39E+03	=	2.68E+03	2.03E+03	+	9.75E+03	6.90E+03	+
f22	2.43E+01	1.13E+01	1.47E+01	3.44E+01	-	3.85E+01	4.65E+01	=	9.71E+00	2.63E+01	-	**7.87E+00**	9.11E+00	-	5.47E+01	7.56E+01	=	6.00E+01	5.56E+01	+	4.27E+01	1.15E+01	+	7.58E+01	3.55E+01	+
f23	**3.23E+02**	4.61E+01	3.29E+02	2.87E-13	=	3.34E+02	4.48E+00	+	3.29E+02	3.72E-03	+	3.29E+02	2.87E-13	=	3.29E+02	2.87E-13	=	3.32E+02	5.68E+00	+	3.31E+02	9.10E-01	+	3.58E+02	1.01E+01	+
f24	**1.25E+02**	7.64E+00	1.35E+02	1.20E+01	+	1.41E+02	1.19E+01	+	1.32E+02	1.12E+01	+	1.34E+02	1.16E+01	+	1.31E+02	1.09E+01	+	1.40E+02	1.16E+01	+	1.46E+02	8.80E+00	+	1.66E+02	9.12E+00	+
f25	1.88E+02	2.69E+01	1.82E+02	2.73E+01	=	1.85E+02	2.73E+01	+	**1.80E+02**	2.96E+01	=	1.86E+02	2.59E+01	+	1.93E+02	2.44E+01	+	1.90E+02	2.60E+01	+	1.85E+02	2.28E+01	=	1.90E+02	1.60E+01	=
f26	**1.00E+02**	7.36E-02	1.00E+02	1.53E-01	+	1.01E+02	2.14E-01	+	1.00E+02	1.14E-01	+	1.00E+02	1.30E-01	+	1.10E+02	2.99E+01	+	1.00E+02	1.69E-01	+	1.00E+02	1.09E-01	+	1.01E+02	1.77E-01	+
f27	2.12E+02	1.99E+02	2.30E+02	1.98E+02	-	1.84E+02	1.96E+02	-	2.41E+02	1.92E+02	-	2.49E+02	1.98E+02	-	3.56E+02	1.35E+02	+	1.76E+02	1.98E+02	-	**2.68E+01**	7.74E+01	=	1.06E+02	1.68E+02	=
f28	**4.13E+02**	5.55E+01	4.71E+02	9.99E+01	+	4.35E+02	5.62E+01	+	4.68E+02	8.43E+01	+	4.72E+02	7.77E+01	+	4.82E+02	9.59E+01	+	4.34E+02	6.30E+01	+	5.74E+02	6.88E+01	+	4.96E+02	8.81E+01	+
f29	7.83E+02	5.90E+02	2.55E+04	1.78E+05	-	9.09E+02	1.32E+03	=	6.82E+04	3.38E+05	-	**5.37E+02**	2.50E+02	-	5.97E+05	8.57E+05	=	9.96E+02	7.73E+02	+	1.40E+04	1.52E+04	+	2.90E+03	2.71E+03	+
f30	**7.70E+02**	1.92E+02	9.47E+02	3.64E+02	+	1.03E+03	3.99E+02	+	8.64E+02	2.73E+02	=	9.52E+02	3.04E+02	+	9.82E+02	4.15E+02	+	1.56E+03	9.36E+02	+	2.18E+03	7.94E+02	+	1.52E+03	1.06E+03	+

Table 12 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2014 in 10 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	1.01E+05	6.17E+04	2.37E+06	2.27E+06	+	1.23E+05	5.06E+04	+	**8.29E+04**	6.14E+04	=
f2	2.47E+03	2.66E+03	7.25E+03	4.19E+03	+	**1.42E+03**	1.41E+03	=	4.48E+03	3.89E+03	+
f3	1.18E+03	9.55E+02	1.27E+04	7.49E+03	+	**7.40E+02**	6.10E+02	-	6.88E+03	5.80E+03	+
f4	**9.25E+00**	1.51E+01	2.05E+01	2.26E+01	+	2.73E+01	1.52E+01	+	1.66E+01	1.67E+01	=
f5	2.00E+01	7.13E-05	2.00E+01	2.77E-03	=	2.00E+01	3.90E-02	+	**1.92E+01**	3.76E+00	=
f6	**1.83E+00**	1.34E+00	1.73E+01	2.57E+00	+	2.02E+00	1.60E+00	=	4.15E+00	1.31E+00	+
f7	1.50E-01	7.14E-02	3.22E+00	6.31E+00	+	2.56E-01	1.17E-01	+	**1.29E-01**	1.13E-01	-
f8	2.12E-13	8.51E-14	1.32E+02	4.73E+01	+	1.38E+01	5.57E+00	+	**1.61E-13**	8.25E-14	-
f9	**1.28E+01**	5.30E+00	1.54E+02	6.46E+01	+	1.50E+01	5.87E+00	=	1.96E+01	6.47E+00	+
f10	1.43E+00	2.08E+00	1.71E+03	4.35E+02	+	2.97E+02	1.90E+02	+	**1.18E+00**	2.19E+00	=
f11	5.42E+02	2.00E+02	1.76E+03	5.02E+02	+	**5.23E+02**	2.45E+02	=	6.53E+02	2.63E+02	+
f12	**7.39E-02**	6.68E-02	3.70E+00	4.39E+00	+	1.38E-01	1.50E-01	+	1.57E-01	1.14E-01	+
f13	1.85E-01	8.27E-02	2.79E-01	1.89E-01	+	**1.85E-01**	8.61E-02	=	4.27E-01	1.99E-01	+
f14	1.92E-01	7.61E-02	6.06E-01	2.92E-01	+	**1.79E-01**	6.45E-02	=	3.63E-01	2.13E-01	+
f15	**1.12E+00**	4.90E-01	3.18E+01	1.80E+01	+	1.22E+00	5.49E-01	=	2.57E+00	1.44E+00	+
f16	2.63E+00	3.80E-01	4.71E+00	2.02E-01	+	**2.49E+00**	4.89E-01	=	3.36E+00	5.80E-01	+
f17	**6.32E+03**	5.19E+03	9.24E+05	1.06E+06	+	7.08E+03	4.96E+03	=	1.48E+05	1.86E+05	+
f18	**5.33E+03**	8.52E+03	1.90E+04	1.45E+04	+	7.80E+03	1.05E+04	=	9.65E+03	1.07E+04	+
f19	**1.48E+00**	6.84E-01	3.26E+01	2.71E+01	+	2.02E+00	5.22E-01	+	1.88E+00	1.13E+00	=
f20	1.18E+03	3.31E+03	1.08E+04	1.13E+04	+	**4.47E+02**	8.59E+02	-	9.70E+03	1.06E+04	+
f21	2.20E+03	2.03E+03	1.86E+06	1.70E+06	+	**1.50E+03**	1.39E+03	-	2.53E+04	2.34E+04	+
f22	**2.43E+01**	1.13E+01	5.74E+02	2.06E+02	+	3.33E+01	1.88E+01	+	9.24E+01	1.12E+02	=
f23	3.23E+02	4.61E+01	3.23E+02	2.58E+01	=	3.29E+02	2.87E-13	=	**3.15E+02**	5.81E+01	=
f24	1.25E+02	7.64E+00	3.30E+02	2.63E+02	+	**1.23E+02**	8.37E+00	=	1.44E+02	1.35E+01	+
f25	1.88E+02	2.69E+01	2.05E+02	4.97E+00	+	1.93E+02	2.33E+01	+	**1.68E+02**	2.85E+01	-
f26	1.00E+02	7.36E-02	1.94E+02	1.02E+02	+	**1.00E+02**	6.17E-02	-	1.20E+02	4.00E+01	+
f27	2.12E+02	1.99E+02	6.14E+02	1.97E+02	+	2.61E+02	1.92E+02	=	**1.73E+01**	5.59E+01	=
f28	**4.13E+02**	5.55E+01	2.70E+03	1.90E+03	+	4.15E+02	6.24E+01	=	5.51E+02	1.10E+02	+
f29	7.83E+02	5.90E+02	1.06E+06	9.44E+05	+	7.84E+04	3.85E+05	+	**5.09E+02**	2.62E+02	-
f30	**7.70E+02**	1.92E+02	2.92E+03	4.15E+03	+	1.08E+03	3.38E+02	+	1.23E+03	4.53E+02	+

Table 13 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2014 in 30 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	**1.02E+06**	4.07E+05	5.80E+06	4.82E+06	+	1.06E+08	1.08E+08	+	9.48E+06	5.77E+06	+	3.12E+06	1.77E+06	+	1.10E+06	6.17E+05	=	8.00E+07	8.41E+07	+	2.20E+08	6.31E+07	+	2.05E+08	9.59E+07	+
f2	**5.87E+03**	8.37E+03	1.06E+04	1.19E+04	+	2.21E+10	6.91E+09	+	4.00E+04	2.47E+04	+	1.44E+04	1.16E+04	+	1.19E+04	1.34E+04	=	1.66E+09	2.73E+09	+	6.70E+09	1.43E+09	+	2.92E+10	6.08E+09	+
f3	**1.05E+03**	5.39E+02	2.02E+04	1.78E+04	+	5.39E+04	1.67E+04	+	8.04E+03	9.95E+03	+	1.85E+04	1.64E+04	+	2.05E+04	1.82E+04	+	7.76E+04	3.66E+04	+	4.32E+04	9.09E+03	+	6.46E+04	1.86E+04	+
f4	5.13E+01	4.03E+01	1.19E+02	3.45E+01	+	2.44E+03	1.20E+03	+	1.08E+02	4.51E+01	+	8.00E+01	4.32E+01	+	**4.68E+01**	4.27E+01	=	5.95E+02	3.80E+02	+	1.27E+03	3.59E+02	+	1.73E+03	1.06E+03	+
f5	**2.00E+01**	6.73E-06	2.00E+01	2.93E-03	+	2.07E+01	1.34E-01	+	2.00E+01	1.82E-02	+	2.00E+01	1.00E-03	+	2.00E+01	4.91E-03	=	2.03E+01	1.91E-01	+	2.10E+01	5.36E-02	+	2.10E+01	8.90E-02	+
f6	**9.90E+00**	3.53E+00	1.73E+01	2.97E+00	+	3.19E+01	2.28E+00	+	1.52E+01	3.36E+00	+	1.67E+01	2.92E+00	+	1.58E+01	3.09E+00	+	2.94E+01	4.20E+00	+	3.04E+01	2.85E+00	+	3.54E+01	3.11E+00	+
f7	**1.12E-02**	1.14E-02	5.72E-02	7.93E-02	+	2.11E+02	5.31E+01	+	1.80E-01	8.36E-02	+	3.69E-02	3.25E-02	+	3.84E-02	4.16E-02	+	1.74E+01	2.23E+01	+	7.68E+01	1.60E+01	+	2.48E+02	5.99E+01	+
f8	**6.11E-13**	1.34E-13	5.37E+00	2.20E+00	+	2.06E+02	3.07E+01	+	7.85E-02	2.43E-01	+	1.96E-02	1.39E-01	+	1.95E-02	1.39E-01	+	1.46E+02	4.22E+01	+	2.28E+02	2.46E+01	+	3.08E+02	2.61E+01	+
f9	8.85E+01	2.15E+01	8.98E+01	2.41E+01	=	2.89E+02	4.14E+01	+	**8.60E+01**	1.99E+01	=	8.84E+01	2.05E+01	=	9.86E+01	2.60E+01	+	2.00E+02	4.43E+01	+	2.75E+02	2.37E+01	+	3.57E+02	2.61E+01	+
f10	1.64E+01	3.21E+01	2.93E+01	3.86E+01	+	4.04E+03	6.46E+02	+	1.33E+01	5.84E+00	+	**1.46E+00**	1.10E+00	-	8.41E+00	3.65E+00	=	4.08E+03	7.86E+02	+	5.53E+03	3.98E+02	+	6.80E+03	5.53E+02	+
f11	2.66E+03	4.51E+02	2.72E+03	5.68E+02	=	5.36E+03	7.17E+02	+	**2.47E+03**	4.87E+02	-	2.53E+03	4.41E+02	=	2.53E+03	4.47E+02	=	4.56E+03	7.28E+02	+	6.99E+03	3.43E+02	+	7.24E+03	4.88E+02	+
f12	1.24E-01	5.28E-02	2.13E-01	8.15E-02	+	9.56E-01	3.62E-01	+	1.74E-01	5.83E-02	+	1.43E-01	4.83E-02	+	**7.97E-02**	2.62E-02	-	9.46E-01	4.60E-01	+	2.72E+00	2.49E-01	+	2.81E+00	3.86E-01	+
f13	**4.56E-01**	1.03E-01	5.26E-01	1.44E-01	+	3.41E+00	4.62E-01	+	6.02E-01	1.43E-01	+	5.76E-01	1.49E-01	+	6.38E-01	1.38E-01	+	7.60E-01	1.62E-01	+	2.07E+00	5.33E-01	+	3.73E+00	7.38E-01	+
f14	**3.42E-01**	2.08E-01	3.55E-01	1.51E-01	+	6.77E+01	1.52E+01	+	3.55E-01	1.39E-01	+	3.67E-01	1.37E-01	+	8.10E-01	3.05E-01	+	5.99E+00	8.02E+00	+	2.78E+01	5.59E+00	+	4.31E+01	1.88E+01	+
f15	**6.19E+00**	1.71E+00	1.47E+01	8.20E+00	+	2.93E+05	3.13E+05	+	1.23E+01	3.54E+00	+	9.42E+00	3.95E+00	+	7.95E+00	2.87E+00	+	1.71E+04	3.57E+04	+	1.02E+03	1.26E+03	+	2.10E+05	1.52E+05	+
f16	1.10E+01	5.57E-01	1.08E+01	6.96E-01	=	1.25E+01	4.45E-01	+	1.04E+01	7.38E-01	-	1.03E+01	6.62E-01	-	**1.01E+01**	7.84E-01	-	1.26E+01	4.47E-01	+	1.29E+01	1.98E-01	+	1.30E+01	2.47E-01	+
f17	**1.37E+05**	6.91E+04	8.16E+05	6.95E+05	+	6.58E+06	4.88E+06	+	1.79E+06	1.07E+06	+	1.16E+06	9.32E+05	+	3.87E+05	2.63E+05	+	3.60E+06	4.10E+06	+	4.72E+06	2.41E+06	+	6.26E+06	3.31E+06	+
f18	8.92E+03	7.04E+03	4.81E+03	5.26E+03	-	3.83E+07	7.49E+07	+	4.91E+03	6.84E+03	-	**3.71E+03**	3.49E+03	-	7.28E+03	7.74E+03	=	8.11E+06	3.37E+07	+	1.69E+07	9.21E+06	+	1.12E+08	8.46E+07	+
f19	**9.35E+00**	1.90E+00	2.12E+01	2.45E+01	+	1.31E+02	3.46E+01	+	2.00E+01	2.60E+01	=	1.66E+01	2.16E+01	-	1.42E+01	1.96E+01	-	9.26E+01	4.19E+01	+	5.58E+01	2.56E+01	+	1.52E+02	5.19E+01	+
f20	**5.90E+02**	3.42E+02	2.50E+04	1.47E+04	+	5.64E+04	3.05E+04	+	5.77E+03	9.15E+03	+	2.35E+04	1.51E+04	+	3.24E+04	1.77E+04	+	4.90E+04	3.75E+04	+	1.98E+04	1.01E+04	+	3.16E+04	1.68E+04	+
f21	**6.63E+04**	3.94E+04	4.33E+05	4.51E+05	+	1.20E+06	9.72E+05	+	7.70E+05	4.74E+05	+	6.56E+05	4.50E+05	+	2.75E+05	2.21E+05	+	8.91E+05	1.02E+06	+	8.71E+05	7.46E+05	+	1.46E+06	1.09E+06	+
f22	**3.80E+02**	1.61E+02	5.95E+02	2.16E+02	+	6.34E+02	2.21E+02	+	5.05E+02	1.97E+02	+	6.26E+02	1.95E+02	+	7.07E+02	2.84E+02	+	5.86E+02	2.13E+02	+	7.78E+02	1.49E+02	+	8.18E+02	2.07E+02	+
f23	3.15E+02	9.02E-04	3.16E+02	1.82E+00	+	4.35E+02	5.12E+01	+	3.16E+02	3.07E-01	+	3.15E+02	7.50E-03	-	**3.15E+02**	5.74E-14	-	3.67E+02	5.11E+01	+	3.47E+02	8.76E+00	+	4.11E+02	5.22E+01	+
f24	**2.27E+02**	1.04E+01	2.37E+02	5.96E+00	+	3.50E+02	1.88E+01	+	2.34E+02	5.54E+00	+	2.36E+02	6.80E+00	+	2.31E+02	8.02E+00	=	3.13E+02	3.21E+01	+	2.68E+02	3.84E+00	+	3.33E+02	1.72E+01	+
f25	**2.06E+02**	1.32E+00	2.12E+02	5.06E+00	+	2.32E+02	1.10E+01	+	2.13E+02	6.86E+00	+	2.10E+02	4.84E+00	+	2.07E+02	2.94E+00	+	2.22E+02	1.16E+01	+	2.34E+02	7.83E+00	+	2.29E+02	7.74E+00	+
f26	**1.00E+02**	1.02E-01	1.01E+02	1.58E-01	+	1.03E+02	5.94E-01	+	1.01E+02	1.29E-01	+	1.28E+02	5.82E+01	+	1.59E+02	7.33E+01	+	1.01E+02	3.87E-01	+	1.02E+02	4.93E-01	+	1.01E+02	3.26E-01	+
f27	**5.00E+02**	1.06E+02	7.22E+02	2.01E+02	+	8.14E+02	2.54E+02	+	7.29E+02	1.83E+02	+	7.14E+02	1.97E+02	+	8.23E+02	1.45E+02	+	5.49E+02	1.81E+02	=	6.57E+02	1.65E+02	+	6.43E+02	2.43E+02	+
f28	**9.46E+02**	8.24E+01	1.32E+03	3.93E+02	+	1.26E+03	1.98E+02	+	1.32E+03	3.85E+02	+	1.31E+03	4.07E+02	+	1.32E+03	4.04E+02	+	1.15E+03	2.35E+02	+	2.77E+03	4.92E+02	+	1.40E+03	2.74E+02	+
f29	5.89E+03	6.28E+03	4.74E+05	1.64E+06	-	1.81E+06	1.98E+06	+	1.71E+05	1.21E+06	-	1.71E+05	1.21E+06	-	8.51E+05	2.60E+06	-	**3.61E+03**	6.08E+03	=	1.87E+07	1.85E+07	+	1.62E+06	2.37E+06	+
f30	4.47E+03	1.75E+03	4.74E+03	2.28E+03	=	8.40E+04	6.41E+04	+	3.90E+03	1.18E+03	=	3.97E+03	2.91E+03	-	**3.41E+03**	1.18E+03	-	1.70E+04	1.63E+04	+	1.02E+05	4.83E+04	+	6.86E+04	4.47E+04	+

Table 14 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2014 in 30 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	1.02E+06	4.07E+05	4.11E+06	2.54E+06	+	5.15E+06	8.76E+05	+	**7.53E+05**	3.98E+05	-
f2	**5.87E+03**	8.37E+03	1.03E+04	9.76E+03	+	7.12E+04	4.31E+04	+	1.33E+04	1.05E+04	+
f3	**1.05E+03**	5.39E+02	4.70E+04	3.28E+04	+	6.50E+03	1.58E+03	+	2.31E+04	1.77E+04	+
f4	5.13E+01	4.03E+01	3.68E+01	4.60E+01	-	6.02E+01	4.20E+01	=	**8.58E+00**	2.22E+01	-
f5	**2.00E+01**	6.73E-06	2.00E+01	1.09E-02	=	2.03E+01	1.60E-01	+	**2.00E+01**	3.66E-04	=
f6	**9.90E+00**	3.53E+00	5.29E+01	4.75E+00	+	1.65E+01	3.55E+00	+	2.14E+01	3.21E+00	+
f7	**1.12E-02**	1.14E-02	1.93E-01	3.95E-01	+	4.34E-01	1.12E-01	+	1.23E-02	1.25E-02	=
f8	6.11E-13	1.34E-13	4.19E+02	7.96E+01	+	7.33E+01	1.68E+01	+	**5.80E-13**	1.86E-13	=
f9	8.85E+01	2.15E+01	6.00E+02	1.07E+02	+	**8.36E+01**	2.30E+01	=	1.83E+02	4.63E+01	+
f10	**1.64E+01**	3.21E+01	5.41E+03	7.30E+02	+	2.62E+03	4.70E+02	+	8.55E+01	1.03E+02	+
f11	**2.66E+03**	4.51E+02	5.60E+03	8.44E+02	+	3.27E+03	6.37E+02	+	3.23E+03	5.70E+02	+
f12	**1.24E-01**	5.28E-02	2.32E+00	1.19E+00	+	4.72E-01	2.56E-01	+	1.83E-01	6.88E-02	+
f13	4.56E-01	1.03E-01	4.74E-01	1.53E-01	=	**3.94E-01**	7.81E-02	-	5.34E-01	1.27E-01	+
f14	3.42E-01	2.08E-01	4.16E-01	1.52E-01	+	**1.80E-01**	3.52E-02	-	3.07E-01	1.13E-01	=
f15	**6.19E+00**	1.71E+00	2.27E+02	9.95E+01	+	8.55E+00	2.47E+00	+	1.13E+01	5.22E+00	+
f16	**1.10E+01**	5.57E-01	1.43E+01	3.98E-01	+	1.14E+01	6.82E-01	+	1.19E+01	6.95E-01	+
f17	**1.37E+05**	6.91E+04	2.62E+06	1.59E+06	+	1.82E+05	7.46E+04	+	4.31E+05	2.69E+05	+
f18	8.92E+03	7.04E+03	6.99E+03	4.87E+03	=	2.04E+04	7.30E+03	+	**3.15E+03**	3.29E+03	-
f19	**9.35E+00**	1.90E+00	3.46E+01	2.65E+01	+	1.37E+01	1.01E+01	+	1.38E+01	2.03E+01	=
f20	**5.90E+02**	3.42E+02	6.74E+04	2.83E+04	+	7.92E+02	3.66E+02	+	4.03E+04	2.19E+04	+
f21	**6.63E+04**	3.94E+04	4.58E+05	3.15E+05	+	7.24E+04	3.98E+04	=	2.68E+05	2.56E+05	+
f22	3.80E+02	1.61E+02	1.24E+03	4.10E+02	+	**3.28E+02**	1.76E+02	=	9.03E+02	2.61E+02	+
f23	3.15E+02	9.02E-04	**3.15E+02**	6.29E-04	-	3.17E+02	1.97E-01	+	**3.15E+02**	5.74E-14	-
f24	2.27E+02	1.04E+01	3.42E+02	1.41E+02	+	**2.11E+02**	1.15E+01	-	2.39E+02	1.77E+01	+
f25	**2.06E+02**	1.32E+00	2.30E+02	2.62E+01	+	2.11E+02	1.48E+00	+	2.17E+02	1.34E+01	+
f26	**1.00E+02**	1.02E-01	2.02E+02	1.40E+02	+	1.01E+02	1.27E-01	+	1.51E+02	5.03E+01	+
f27	**5.00E+02**	1.06E+02	1.45E+03	6.44E+02	+	6.08E+02	1.16E+02	+	6.28E+02	2.59E+02	=
f28	**9.46E+02**	8.24E+01	6.46E+03	1.95E+03	+	1.39E+03	3.27E+02	+	2.86E+03	8.45E+02	+
f29	**5.89E+03**	6.28E+03	1.85E+06	3.56E+06	=	6.12E+03	2.29E+03	+	1.50E+06	3.27E+06	-
f30	4.47E+03	1.75E+03	4.51E+03	3.59E+03	=	1.05E+04	2.05E+03	+	**3.06E+03**	1.08E+03	-

Table 15 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2014 in 50 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	1.82E+06	6.40E+05	3.45E+07	1.94E+07	+	1.72E+08	2.37E+08	+	1.21E+07	4.56E+06	+	3.72E+06	1.65E+06	+	**1.61E+06**	5.92E+05	=	1.48E+08	9.23E+07	+	7.04E+08	2.63E+08	+	4.79E+08	1.38E+08	+
f2	**1.95E+03**	3.10E+03	1.47E+07	3.22E+07	+	1.50E+10	1.01E+10	+	9.38E+04	7.03E+04	+	6.78E+03	8.49E+03	+	9.80E+03	1.15E+04	+	1.58E+10	1.09E+10	+	4.43E+10	6.69E+09	+	8.54E+10	1.60E+10	+
f3	**3.56E+02**	1.71E+02	3.26E+04	1.48E+04	+	1.04E+05	5.11E+04	+	1.93E+04	1.01E+04	+	3.29E+04	1.63E+04	+	3.26E+04	1.36E+04	+	1.47E+05	3.99E+04	+	1.34E+05	2.23E+04	+	1.27E+05	2.81E+04	+
f4	9.06E+01	1.90E+01	2.35E+02	5.61E+01	+	3.80E+03	1.91E+03	+	1.27E+02	3.70E+01	+	9.43E+01	3.28E+01	=	**8.40E+01**	2.98E+01	-	1.78E+03	1.15E+03	+	8.35E+03	3.20E+03	+	1.04E+04	5.55E+03	+
f5	**2.00E+01**	1.06E-03	2.00E+01	2.60E-02	+	2.07E+01	1.61E-01	+	2.01E+01	2.59E-02	+	2.00E+01	7.55E-04	+	2.00E+01	5.30E-03	-	2.04E+01	1.63E-01	+	2.12E+01	3.21E-02	+	2.12E+01	3.75E-02	+
f6	**2.32E+01**	4.93E+00	3.64E+01	3.53E+00	+	6.03E+01	3.59E+00	+	3.24E+01	4.39E+00	+	3.24E+01	3.90E+00	+	3.00E+01	3.95E+00	+	5.38E+01	6.09E+00	+	6.13E+01	3.79E+00	+	6.49E+01	3.82E+00	+
f7	**4.01E-03**	6.05E-03	1.30E+00	5.20E-01	+	8.05E+02	1.74E+02	+	3.33E-01	1.35E-01	+	2.47E-02	3.01E-02	+	1.23E-02	1.12E-02	+	1.62E+02	9.32E+01	+	3.94E+02	4.73E+01	+	8.95E+02	1.57E+02	+
f8	1.95E-02	1.39E-01	2.26E+01	4.92E+00	+	5.10E+02	4.82E+01	+	7.03E-01	7.93E-01	+	5.85E-02	2.36E-01	+	**6.99E-12**	1.96E-12	+	3.38E+02	8.56E+01	+	5.11E+02	3.06E+01	+	6.28E+02	4.33E+01	+
f9	1.84E+02	3.00E+01	1.94E+02	3.74E+01	=	6.77E+02	6.44E+01	+	1.87E+02	4.01E+01	=	**1.70E+02**	4.02E+01	=	2.17E+02	5.35E+01	+	4.60E+02	8.37E+01	+	5.77E+02	3.12E+01	+	7.37E+02	5.43E+01	+
f10	2.50E+01	4.23E+01	2.41E+02	1.77E+02	+	8.91E+03	1.10E+03	+	1.19E+02	1.01E+02	+	**1.97E+00**	1.27E+00	-	1.26E+01	1.66E+01	=	8.17E+03	1.24E+03	+	1.13E+04	4.86E+02	+	1.31E+04	7.05E+02	+
f11	5.24E+03	6.39E+02	5.27E+03	7.37E+02	=	1.12E+04	8.25E+02	+	5.09E+03	7.80E+02	=	5.00E+03	7.10E+02	=	**4.72E+03**	6.33E+02	-	9.01E+03	1.11E+03	+	1.33E+04	4.31E+02	+	1.37E+04	4.51E+02	+
f12	1.20E-01	3.14E-02	2.53E-01	5.69E-02	+	1.49E+00	3.88E-01	+	1.76E-01	5.09E-02	+	1.26E-01	3.79E-02	=	**9.16E-02**	3.16E-02	-	1.15E+00	4.52E-01	+	3.55E+00	2.86E-01	+	3.77E+00	4.13E-01	+
f13	**6.08E-01**	1.12E-01	6.49E-01	1.21E-01	=	5.71E+00	6.47E-01	+	6.63E-01	1.28E-01	+	6.36E-01	1.25E-01	=	6.82E-01	1.11E-01	+	1.64E+00	1.20E+00	+	4.41E+00	3.28E-01	+	5.91E+00	6.85E-01	+
f14	**3.87E-01**	2.41E-01	5.40E-01	2.90E-01	+	2.20E+02	3.78E+01	+	6.17E-01	3.31E-01	+	4.53E-01	2.11E-01	+	7.29E-01	3.46E-01	+	4.29E+01	2.27E+01	+	1.01E+02	1.67E+01	+	2.48E+02	3.97E+01	+
f15	**1.29E+01**	3.25E+00	6.24E+01	3.58E+01	+	5.56E+06	4.05E+06	+	3.02E+01	9.12E+00	+	1.91E+01	6.02E+00	+	1.95E+01	6.99E+00	+	2.84E+05	5.33E+05	+	1.54E+05	1.03E+05	+	2.79E+06	1.52E+06	+
f16	1.95E+01	8.14E-01	1.96E+01	7.58E-01	=	2.18E+01	4.44E-01	+	1.89E+01	8.61E-01	-	**1.84E+01**	8.36E-01	-	1.88E+01	7.94E-01	-	2.20E+01	5.12E-01	+	2.26E+01	2.66E-01	+	2.26E+01	3.31E-01	+
f17	**1.84E+05**	1.03E+05	3.37E+06	3.03E+06	+	5.96E+07	3.91E+07	+	4.19E+06	2.31E+06	+	1.29E+06	7.89E+05	+	3.27E+05	1.97E+05	+	1.48E+07	1.08E+07	+	3.49E+07	2.41E+07	+	2.77E+07	1.35E+07	+
f18	**1.59E+03**	1.27E+03	1.92E+03	1.25E+03	=	1.57E+09	1.09E+09	+	4.50E+03	4.14E+03	+	2.09E+03	1.85E+03	=	2.61E+03	1.95E+03	+	2.01E+07	5.81E+07	+	2.71E+08	1.27E+08	+	4.56E+08	2.77E+08	+
f19	2.15E+01	6.12E+00	7.21E+01	2.13E+01	+	4.19E+02	1.70E+02	+	4.91E+01	2.51E+01	+	2.18E+01	1.32E+01	-	**2.09E+01**	1.43E+01	-	1.93E+02	1.20E+02	+	2.05E+02	3.84E+01	+	2.18E+02	4.73E+01	+
f20	**7.86E+02**	2.38E+02	6.13E+04	3.25E+04	+	1.90E+05	1.01E+05	+	5.48E+03	4.94E+03	+	6.07E+04	2.75E+04	+	6.74E+04	2.64E+04	+	1.17E+05	7.28E+04	+	5.62E+04	2.58E+04	+	8.25E+04	5.62E+04	+
f21	**2.13E+05**	9.71E+04	2.05E+06	2.38E+06	+	1.85E+07	9.89E+06	+	3.09E+06	1.18E+06	+	1.07E+06	6.71E+05	+	3.29E+05	1.93E+05	+	7.59E+06	5.43E+06	+	8.19E+06	4.58E+06	+	1.18E+07	6.99E+06	+
f22	**8.05E+02**	2.85E+02	1.22E+03	3.65E+02	+	1.72E+03	3.38E+02	+	1.25E+03	3.65E+02	+	1.29E+03	2.94E+02	+	1.42E+03	3.80E+02	+	1.34E+03	3.77E+02	+	2.38E+03	3.09E+02	+	2.03E+03	2.71E+02	+
f23	3.44E+02	2.47E-01	3.53E+02	4.05E+00	+	8.40E+02	1.93E+02	+	3.44E+02	1.89E-02	-	3.44E+02	4.15E-02	-	**3.44E+02**	4.59E-13	-	4.30E+02	6.32E+01	+	4.68E+02	2.58E+01	+	1.09E+03	1.67E+02	+
f24	2.70E+02	5.14E+00	2.80E+02	4.05E+00	+	5.60E+02	3.24E+01	+	2.68E+02	6.56E+00	=	2.66E+02	5.72E+00	-	**2.63E+02**	6.19E+00	-	4.73E+02	5.27E+01	+	3.71E+02	9.97E+00	+	5.39E+02	3.70E+01	+
f25	2.13E+02	2.71E+00	2.24E+02	5.57E+00	+	3.19E+02	2.92E+01	+	2.20E+02	5.06E+00	+	2.16E+02	5.20E+00	+	**2.12E+02**	4.17E+00	=	2.56E+02	1.86E+01	+	2.91E+02	1.63E+01	+	2.82E+02	2.27E+01	+
f26	1.07E+02	2.38E+01	1.57E+02	6.88E+01	+	1.54E+02	8.42E+01	+	1.52E+02	5.08E+01	+	1.65E+02	7.84E+01	+	1.40E+02	5.90E+01	+	**1.04E+02**	1.68E+01	+	1.28E+02	5.40E+01	+	1.43E+02	6.87E+01	+
f27	**8.87E+02**	1.33E+02	1.32E+03	1.13E+02	+	1.86E+03	9.74E+01	+	1.26E+03	1.21E+02	+	1.22E+03	1.52E+02	+	1.26E+03	1.11E+02	+	1.64E+03	1.35E+02	+	2.09E+03	8.63E+01	+	1.85E+03	1.02E+02	+
f28	**1.66E+03**	3.26E+02	2.61E+03	9.01E+02	+	3.14E+03	9.75E+02	+	2.31E+03	7.83E+02	+	2.47E+03	7.96E+02	+	2.47E+03	7.83E+02	+	2.35E+03	7.22E+02	+	8.22E+03	1.02E+03	+	2.51E+03	7.05E+02	+
f29	1.19E+04	1.63E+04	6.99E+05	4.93E+06	+	4.99E+07	2.01E+07	+	**2.60E+03**	1.30E+03	-	6.93E+05	4.93E+06	-	5.57E+06	1.30E+07	-	3.46E+05	1.89E+06	+	2.57E+08	1.40E+08	+	2.23E+07	1.26E+07	+
f30	1.57E+04	4.39E+03	1.84E+04	5.22E+03	+	6.67E+05	5.86E+05	+	1.44E+04	2.90E+03	=	1.28E+04	2.73E+03	-	**1.22E+04**	2.19E+03	-	6.15E+04	4.27E+04	+	1.26E+06	4.87E+05	+	2.25E+05	1.26E+05	+

Table 16 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2014 in 50 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	1.82E+06	6.40E+05	3.63E+06	1.67E+06	+	5.51E+06	6.45E+05	+	**1.54E+06**	6.05E+05	-
f2	**1.95E+03**	3.10E+03	1.20E+04	1.10E+04	+	9.54E+07	2.39E+07	+	6.70E+03	7.89E+03	+
f3	**3.56E+02**	1.71E+02	4.30E+04	1.58E+04	+	1.40E+04	1.86E+03	+	2.89E+04	1.20E+04	+
f4	9.06E+01	1.90E+01	8.21E+01	3.64E+01	-	2.10E+02	4.29E+01	+	**5.04E+01**	4.41E+01	-
f5	**2.00E+01**	1.06E-03	2.00E+01	6.18E-03	=	2.06E+01	1.19E-01	+	**2.00E+01**	2.21E-03	=
f6	**2.32E+01**	4.93E+00	8.91E+01	6.59E+00	+	3.60E+01	5.40E+00	+	4.12E+01	4.55E+00	+
f7	**4.01E-03**	6.05E-03	6.24E-02	9.73E-02	+	3.68E+00	4.47E-01	+	1.12E-02	1.05E-02	+
f8	1.95E-02	1.39E-01	7.06E+02	1.05E+02	+	1.60E+02	2.34E+01	+	**9.47E-13**	1.92E-13	-
f9	1.84E+02	3.00E+01	1.11E+03	2.19E+02	+	**1.65E+02**	3.50E+01	-	3.82E+02	8.99E+01	+
f10	**2.50E+01**	4.23E+01	8.49E+03	1.04E+03	+	5.32E+03	8.51E+02	+	3.48E+02	1.85E+02	+
f11	**5.24E+03**	6.39E+02	8.95E+03	1.01E+03	+	6.99E+03	1.40E+03	+	5.84E+03	8.97E+02	+
f12	**1.20E-01**	3.14E-02	2.14E+00	6.83E-01	+	7.77E-01	3.60E-01	+	1.62E-01	5.34E-02	+
f13	6.08E-01	1.12E-01	**4.62E-01**	1.15E-01	-	6.45E-01	8.90E-02	=	5.85E-01	1.17E-01	=
f14	3.87E-01	2.41E-01	4.76E-01	1.71E-01	+	**2.34E-01**	2.41E-02	-	3.97E-01	1.86E-01	+
f15	**1.29E+01**	3.25E+00	4.87E+02	1.32E+02	+	3.48E+01	6.70E+00	+	2.00E+01	6.97E+00	+
f16	**1.95E+01**	8.14E-01	2.38E+01	5.72E-01	+	2.07E+01	7.30E-01	+	2.05E+01	7.00E-01	+
f17	**1.84E+05**	1.03E+05	1.65E+06	6.96E+05	+	2.28E+05	6.56E+04	+	3.66E+05	1.65E+05	+
f18	**1.59E+03**	1.27E+03	3.24E+03	1.86E+03	+	1.60E+03	1.10E+03	=	2.27E+03	1.64E+03	=
f19	2.15E+01	6.12E+00	3.26E+01	1.05E+01	+	4.72E+01	2.82E+01	+	**1.71E+01**	2.18E+00	-
f20	**7.86E+02**	2.38E+02	9.57E+04	3.92E+04	+	9.74E+02	3.07E+02	+	5.49E+04	2.70E+04	+
f21	2.13E+05	9.71E+04	1.19E+06	8.74E+05	+	**1.63E+05**	5.35E+04	-	4.69E+05	3.22E+05	+
f22	**8.05E+02**	2.85E+02	2.05E+03	5.99E+02	+	9.07E+02	2.71E+02	=	1.60E+03	4.28E+02	+
f23	3.44E+02	2.47E-01	**3.44E+02**	4.59E-13	-	3.63E+02	8.83E-01	+	**3.44E+02**	4.59E-13	-
f24	2.70E+02	5.14E+00	3.95E+02	1.48E+02	+	**2.58E+02**	1.25E+01	-	2.81E+02	5.10E+01	=
f25	**2.13E+02**	2.71E+00	2.42E+02	2.99E+01	+	2.25E+02	1.79E+00	+	2.19E+02	9.08E+00	+
f26	1.07E+02	2.38E+01	2.41E+02	1.72E+02	+	**1.05E+02**	1.95E+01	+	1.51E+02	5.04E+01	=
f27	**8.87E+02**	1.33E+02	2.81E+03	2.47E+02	+	1.26E+03	1.37E+02	+	1.47E+03	2.45E+02	+
f28	**1.66E+03**	3.26E+02	1.24E+04	2.85E+03	+	3.87E+03	8.59E+02	+	5.20E+03	1.77E+03	+
f29	**1.19E+04**	1.63E+04	1.24E+07	1.70E+07	=	7.82E+06	3.92E+07	+	1.38E+06	6.91E+06	-
f30	1.57E+04	4.39E+03	1.62E+04	3.90E+03	=	7.56E+04	8.45E+03	+	**1.26E+04**	2.45E+03	-

Table 17 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2014 in 100 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	5.51E+06	1.31E+06	2.93E+08	9.23E+07	+	5.90E+08	3.83E+08	+	8.47E+07	2.72E+07	+	1.43E+07	6.63E+06	+	**4.17E+06**	1.24E+06	-	1.05E+09	5.26E+08	+	2.73E+09	1.11E+09	+	2.28E+09	5.55E+08	+
f2	**1.27E+04**	1.63E+04	5.17E+09	1.99E+09	+	1.79E+10	3.57E+10	+	5.88E+05	3.59E+05	+	2.49E+04	3.17E+04	+	3.98E+04	4.49E+04	+	1.03E+11	4.22E+10	+	1.86E+11	1.67E+10	+	3.11E+11	3.05E+10	+
f3	**1.29E+02**	6.01E+01	5.86E+04	1.82E+04	+	2.58E+05	9.49E+04	+	2.61E+04	1.48E+04	+	5.71E+04	2.54E+04	+	5.05E+04	1.72E+04	+	3.84E+05	7.34E+04	+	3.75E+05	3.81E+04	+	2.81E+05	4.92E+04	+
f4	2.07E+02	3.12E+01	1.17E+03	3.49E+02	+	4.66E+03	4.21E+03	+	2.61E+02	4.80E+01	+	1.97E+02	3.52E+01	=	**1.75E+02**	2.95E+01	-	1.50E+04	8.51E+03	+	4.10E+04	1.09E+04	+	7.18E+04	1.44E+04	+
f5	**2.00E+01**	0.00E+00	2.04E+01	6.52E-02	+	2.11E+01	1.71E-01	+	2.02E+01	4.86E-02	+	2.00E+01	1.55E-03	+	2.00E+01	4.07E-03	-	2.07E+01	1.27E-01	+	2.13E+01	2.78E-02	+	2.14E+01	2.97E-02	+
f6	**6.31E+01**	7.99E+00	9.65E+01	7.43E+00	+	1.41E+02	4.93E+00	+	8.52E+01	6.55E+00	+	7.95E+01	7.09E+00	+	8.37E+01	5.24E+00	+	1.24E+02	8.67E+00	+	1.51E+02	4.02E+00	+	1.47E+02	6.27E+00	+
f7	**2.03E-03**	4.14E-03	4.67E+01	1.72E+01	+	2.80E+03	3.52E+02	+	4.50E-01	1.31E-01	+	1.05E-02	1.26E-02	+	4.35E-03	6.81E-03	-	8.49E+02	3.64E+02	+	1.75E+03	2.27E+02	+	2.84E+03	3.18E+02	+
f8	5.85E-02	2.36E-01	1.54E+02	2.38E+01	+	1.39E+03	7.11E+01	+	8.35E+00	3.58E+00	+	**1.95E-02**	1.39E-01	+	3.90E-02	1.95E-01	+	9.80E+02	1.61E+02	+	1.28E+03	5.11E+01	+	1.51E+03	5.64E+01	+
f9	**5.03E+02**	6.88E+01	7.20E+02	1.05E+02	+	1.63E+03	1.30E+02	+	5.73E+02	9.16E+01	+	5.04E+02	7.58E+01	=	6.42E+02	9.74E+01	+	1.23E+03	1.86E+02	+	1.46E+03	5.82E+01	+	1.73E+03	7.68E+01	+
f10	6.68E+01	7.62E+01	3.12E+03	5.82E+02	+	2.32E+04	1.29E+03	+	9.50E+02	3.98E+02	+	**3.42E+00**	1.10E+00	-	1.76E+01	2.32E+01	=	2.04E+04	2.34E+03	+	2.76E+04	6.16E+02	+	3.05E+04	6.48E+02	+
f11	**1.21E+04**	1.06E+03	1.59E+04	1.11E+03	+	2.67E+04	1.19E+03	+	1.24E+04	1.37E+03	=	1.23E+04	1.15E+03	=	1.26E+04	1.21E+03	=	2.17E+04	2.11E+03	+	3.05E+04	5.30E+02	+	3.07E+04	6.09E+02	+
f12	2.16E-01	4.30E-02	5.36E-01	8.83E-02	+	2.15E+00	3.59E-01	+	3.14E-01	6.94E-02	+	1.97E-01	4.48E-02	-	**1.54E-01**	4.01E-02	-	1.37E+00	5.11E-01	+	4.16E+00	2.31E-01	+	4.32E+00	2.77E-01	+
f13	6.46E-01	6.89E-02	6.94E-01	9.11E-02	+	9.18E+00	6.92E-01	+	7.11E-01	9.30E-02	+	**6.28E-01**	7.90E-02	=	6.83E-01	8.51E-02	+	4.67E+00	1.09E+00	+	7.15E+00	4.71E-01	+	9.26E+00	5.79E-01	+
f14	2.06E-01	1.81E-02	1.47E-01	1.84E-02	-	2.78E+02	3.76E+01	+	2.41E-01	2.40E-02	+	**1.46E-01**	1.88E-02	-	1.52E-01	1.72E-02	-	4.67E+01	2.76E+01	+	1.83E+02	3.77E+01	+	1.45E+02	6.59E+01	+
f15	**3.43E+01**	6.79E+00	6.18E+03	7.29E+03	+	5.30E+07	1.90E+07	+	9.25E+01	1.55E+01	+	4.91E+01	1.11E+01	+	5.29E+01	1.24E+01	+	5.38E+06	7.57E+06	+	5.26E+06	1.98E+06	+	3.45E+07	1.37E+07	+
f16	4.18E+01	9.08E-01	4.33E+01	8.06E-01	+	4.59E+01	5.48E-01	+	4.25E+01	8.69E-01	+	**4.04E+01**	1.14E+00	-	4.11E+01	1.18E+00	-	4.56E+01	8.53E-01	+	4.69E+01	2.63E-01	+	4.68E+01	3.37E-01	+
f17	**5.83E+05**	2.06E+05	4.76E+07	2.23E+07	+	4.24E+08	1.22E+08	+	8.59E+06	3.26E+06	+	2.78E+06	1.09E+06	+	8.65E+05	2.81E+05	+	1.11E+08	5.90E+07	+	2.89E+08	1.04E+08	+	1.92E+08	6.32E+07	+
f18	**2.66E+03**	2.11E+03	2.94E+07	5.49E+07	+	1.65E+10	4.93E+09	+	1.57E+05	4.76E+05	+	3.30E+03	3.85E+03	=	4.08E+03	3.50E+03	=	1.61E+09	1.70E+09	+	3.40E+09	8.54E+08	+	1.47E+10	4.37E+09	+
f19	1.07E+02	1.68E+01	2.04E+02	4.23E+01	+	2.09E+03	5.74E+02	+	1.18E+02	2.08E+01	+	**1.01E+02**	2.27E+01	-	1.02E+02	1.78E+01	-	7.04E+02	2.50E+02	+	1.47E+03	5.69E+02	+	9.77E+02	4.29E+02	+
f20	**1.18E+03**	2.22E+02	1.63E+05	4.64E+04	+	1.58E+06	1.06E+06	+	1.75E+04	7.71E+03	+	1.07E+05	3.21E+04	+	1.46E+05	4.85E+04	+	4.23E+05	3.07E+05	+	2.71E+05	1.25E+05	+	2.87E+05	2.17E+05	+
f21	**3.54E+05**	1.49E+05	1.82E+07	7.32E+06	+	1.71E+08	6.90E+07	+	8.09E+06	2.55E+06	+	2.26E+06	9.88E+05	+	5.22E+05	2.43E+05	+	3.99E+07	2.15E+07	+	1.08E+08	4.02E+07	+	8.16E+07	2.88E+07	+
f22	**1.90E+03**	5.15E+02	3.01E+03	6.12E+02	+	1.04E+04	6.13E+03	+	2.83E+03	5.90E+02	+	2.80E+03	5.14E+02	+	3.18E+03	5.89E+02	+	4.20E+03	1.54E+03	+	5.85E+03	4.30E+02	+	7.43E+03	1.10E+03	+
f23	3.51E+02	2.01E+00	5.56E+02	9.23E+01	+	2.69E+03	5.46E+02	+	3.50E+02	1.55E+00	-	3.50E+02	1.60E+00	-	**3.48E+02**	1.33E-01	-	1.12E+03	3.85E+02	+	8.63E+02	6.54E+01	+	2.86E+03	3.27E+02	+
f24	3.57E+02	8.26E+00	4.59E+02	1.38E+01	+	1.25E+03	7.18E+01	+	3.63E+02	6.78E+00	+	3.51E+02	9.42E+00	-	**3.48E+02**	1.20E+01	-	9.60E+02	1.02E+02	+	7.62E+02	2.70E+01	+	1.14E+03	8.20E+01	+
f25	**2.42E+02**	5.63E+00	3.00E+02	1.69E+01	+	7.01E+02	1.03E+02	+	2.61E+02	1.10E+01	+	2.57E+02	1.32E+01	+	2.46E+02	1.25E+01	=	4.04E+02	5.34E+01	+	6.12E+02	6.04E+01	+	5.24E+02	9.16E+01	+
f26	**1.56E+02**	5.10E+01	2.25E+02	7.18E+00	+	5.60E+02	1.60E+02	+	2.00E+02	2.04E+01	+	1.99E+02	1.41E+01	+	1.91E+02	3.00E+01	=	2.87E+02	7.54E+01	+	4.23E+02	8.22E+01	+	4.69E+02	1.27E+02	+
f27	**1.81E+03**	2.68E+02	2.83E+03	1.61E+02	+	3.94E+03	1.46E+02	+	2.53E+03	1.83E+02	+	2.38E+03	1.50E+02	+	2.53E+03	2.30E+02	+	3.45E+03	2.22E+02	+	4.56E+03	1.26E+02	+	4.00E+03	1.49E+02	+
f28	**4.80E+03**	1.12E+03	6.85E+03	1.46E+03	+	1.05E+04	1.28E+03	+	5.71E+03	1.08E+03	+	5.60E+03	9.15E+02	+	5.83E+03	1.02E+03	+	5.94E+03	1.04E+03	+	2.42E+04	1.49E+03	+	6.94E+03	1.11E+03	+
f29	9.53E+03	1.32E+04	5.77E+05	5.10E+05	+	4.01E+08	1.09E+08	+	5.47E+03	7.86E+02	=	**4.32E+03**	5.51E+02	-	4.90E+06	1.98E+07	-	1.84E+07	2.53E+07	+	1.76E+09	4.22E+08	+	9.50E+07	1.02E+07	+
f30	3.57E+04	1.54E+04	2.60E+05	1.39E+05	+	2.22E+07	8.98E+06	+	4.22E+04	1.77E+04	=	2.32E+04	5.39E+03	-	**1.06E+04**	9.94E+02	-	2.75E+06	3.50E+06	+	1.22E+07	5.35E+06	+	4.46E+06	4.72E+06	+

Table 18 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2014 in 100 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	**5.51E+06**	1.31E+06	1.46E+07	6.56E+06	+	4.56E+07	2.06E+06	+	1.17E+07	3.44E+06	+
f2	**1.27E+04**	1.63E+04	2.24E+04	2.86E+04	+	8.20E+09	4.63E+08	+	2.65E+04	3.31E+04	+
f3	**1.29E+02**	6.01E+01	6.66E+04	2.53E+04	+	5.30E+04	3.10E+03	+	4.50E+04	1.71E+04	+
f4	2.07E+02	3.12E+01	**1.66E+02**	3.25E+01	-	6.97E+02	6.46E+01	+	1.94E+02	4.40E+01	=
f5	**2.00E+01**	0.00E+00	**2.00E+01**	2.87E-04	=	2.09E+01	1.24E-01	+	**2.00E+01**	2.57E-03	=
f6	**6.31E+01**	7.99E+00	1.82E+02	8.70E+00	+	9.62E+01	7.27E+00	+	1.01E+02	7.17E+00	+
f7	**2.03E-03**	4.14E-03	2.92E-02	5.08E-02	=	1.03E+02	7.97E+00	+	7.00E-03	8.59E-03	=
f8	5.85E-02	2.36E-01	1.40E+03	1.55E+02	+	4.12E+02	4.28E+01	+	**2.00E-12**	2.79E-13	-
f9	5.03E+02	6.88E+01	2.03E+03	2.70E+02	+	**4.43E+02**	5.17E+01	-	1.13E+03	1.74E+02	+
f10	**6.68E+01**	7.62E+01	1.75E+04	1.23E+03	+	1.38E+04	1.59E+03	+	1.81E+03	3.21E+02	+
f11	**1.21E+04**	1.06E+03	1.83E+04	1.70E+03	+	1.82E+04	3.53E+03	+	1.47E+04	1.33E+03	+
f12	**2.16E-01**	4.30E-02	2.13E+00	6.68E-01	+	1.30E+00	3.40E-01	+	2.24E-01	5.78E-02	=
f13	6.46E-01	6.89E-02	**5.37E-01**	1.17E-01	-	6.17E-01	5.50E-02	-	6.13E-01	6.81E-02	-
f14	2.06E-01	1.81E-02	1.79E-01	2.05E-02	-	**1.25E-01**	1.20E-02	-	1.55E-01	2.17E-02	-
f15	**3.43E+01**	6.79E+00	1.32E+03	2.51E+02	+	3.27E+02	4.16E+01	+	5.13E+01	9.66E+00	+
f16	**4.18E+01**	9.08E-01	4.73E+01	8.60E-01	+	4.38E+01	1.05E+00	+	4.36E+01	8.21E-01	+
f17	**5.83E+05**	2.06E+05	4.23E+06	1.72E+06	+	1.37E+06	1.94E+05	+	3.03E+06	1.45E+06	+
f18	2.66E+03	2.11E+03	4.92E+03	5.13E+03	=	**1.22E+03**	5.35E+02	-	3.12E+03	2.53E+03	=
f19	1.07E+02	1.68E+01	1.18E+02	2.84E+01	+	1.15E+02	3.59E+01	=	**8.49E+01**	3.18E+01	-
f20	**1.18E+03**	2.22E+02	2.28E+05	7.08E+04	+	8.63E+03	1.34E+03	+	4.38E+04	2.16E+04	+
f21	**3.54E+05**	1.49E+05	3.38E+06	1.71E+06	+	8.40E+05	1.33E+05	+	3.47E+06	2.01E+06	+
f22	**1.90E+03**	5.15E+02	4.12E+03	6.99E+02	+	1.94E+03	4.40E+02	=	2.97E+03	6.04E+02	+
f23	3.51E+02	2.01E+00	3.49E+02	1.12E+00	-	4.25E+02	2.17E+00	+	**3.49E+02**	4.20E-01	-
f24	3.57E+02	8.26E+00	5.44E+02	5.36E+01	+	**3.36E+02**	3.43E+01	-	3.60E+02	7.27E+00	=
f25	2.42E+02	5.63E+00	2.96E+02	5.25E+01	+	**2.01E+02**	7.42E+00	-	2.81E+02	2.69E+01	+
f26	**1.56E+02**	5.10E+01	1.95E+02	2.39E+01	=	2.48E+02	1.57E+02	=	1.99E+02	1.41E+01	+
f27	**1.81E+03**	2.68E+02	5.22E+03	1.26E+03	+	2.88E+03	2.54E+02	+	2.96E+03	2.23E+02	+
f28	**4.80E+03**	1.12E+03	3.06E+04	5.47E+03	+	1.46E+04	1.27E+03	+	1.25E+04	2.66E+03	+
f29	9.53E+03	1.32E+04	2.61E+07	3.90E+07	=	2.72E+07	1.94E+08	+	**3.86E+03**	9.35E+02	-
f30	3.57E+04	1.54E+04	1.34E+04	2.13E+03	-	1.60E+05	1.42E+04	+	**1.01E+04**	9.33E+02	-

Table 19 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2017 in 10 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	**2.19E+03**	2.39E+03	3.86E+03	3.39E+03	+	8.37E+07	1.50E+08	+	9.07E+03	9.39E+03	+	2.95E+03	2.91E+03	=	4.93E+03	4.00E+03	+	3.33E+06	2.37E+07	+	9.24E+07	2.79E+07	+	2.85E+08	1.25E+08	+
f2	1.94E+01	3.33E+01	1.31E+03	3.94E+03	+	8.95E+07	2.61E+08	+	1.76E-01	4.78E-01	-	1.65E+03	3.05E+03	+	**4.16E-04**	7.20E-04	=	2.75E+08	7.95E+08	+	2.04E+05	3.02E+05	+	2.00E+07	4.45E+07	+
f3	1.38E-12	9.10E-13	1.21E+00	5.59E+00	+	4.26E+03	3.58E+03	+	2.54E+02	2.85E+02	+	2.88E+00	1.29E+01	+	**8.19E-13**	1.45E-12	-	6.41E+03	5.95E+03	+	8.93E+02	4.93E+02	+	2.59E+03	1.80E+03	+
f4	**6.32E-01**	3.23E-01	1.01E+01	2.01E+01	+	4.14E+01	3.19E+01	+	1.07E+01	1.93E+01	+	5.93E+00	1.23E+01	+	3.27E+00	1.38E+01	-	2.64E+01	3.80E+01	+	1.79E+01	1.73E+01	+	3.06E+01	2.50E+01	+
f5	1.46E+01	4.84E+00	1.33E+01	5.50E+00	=	2.62E+01	1.08E+01	+	**1.18E+01**	4.73E+00	-	1.36E+01	4.83E+00	=	1.37E+01	5.07E+00	=	2.21E+01	8.32E+00	+	4.02E+01	7.78E+00	+	4.86E+01	8.34E+00	+
f6	5.44E-04	1.93E-03	9.88E-03	1.94E-02	+	9.72E+00	4.53E+00	+	6.33E-02	3.33E-02	+	5.79E-02	6.94E-02	+	**7.49E-07**	2.05E-06	-	1.59E+01	7.95E+00	+	5.52E+00	2.20E+00	+	2.34E+01	7.91E+00	+
f7	2.35E+01	7.14E+00	2.71E+01	8.27E+00	=	6.51E+01	2.18E+01	+	2.64E+01	7.40E+00	=	2.61E+01	9.35E+00	=	**1.81E+01**	3.76E+00	-	8.68E+01	4.37E+01	+	6.63E+01	8.76E+00	+	1.15E+02	1.74E+01	+
f8	1.57E+01	7.46E+00	1.55E+01	6.72E+00	=	2.95E+01	1.18E+01	+	1.45E+01	5.91E+00	=	1.67E+01	6.36E+00	=	**1.29E+01**	5.85E+00	-	3.63E+01	1.44E+01	+	3.52E+01	8.44E+00	+	5.86E+01	9.84E+00	+
f9	**3.56E-02**	2.42E-01	8.87E+00	1.10E+01	+	2.27E+02	1.49E+02	+	5.14E+00	9.68E+00	+	1.38E+01	2.30E+01	+	3.01E+00	2.15E+01	-	5.02E+02	3.48E+02	+	1.77E+01	1.36E+01	+	4.17E+02	2.28E+02	+
f10	5.92E+02	2.36E+02	5.11E+02	2.30E+02	-	7.22E+02	2.47E+02	+	5.26E+02	2.14E+02	=	5.79E+02	1.84E+02	=	**4.52E+02**	2.14E+02	-	9.09E+02	2.62E+02	+	1.27E+03	1.99E+02	+	1.50E+03	2.39E+02	+
f11	1.32E+01	6.38E+00	9.83E+00	4.61E+00	-	6.84E+01	9.48E+01	+	1.08E+01	4.85E+00	=	1.11E+01	6.74E+00	=	**7.12E+00**	4.52E+00	-	1.04E+02	7.63E+01	+	3.56E+01	9.24E+00	+	2.07E+02	7.82E+01	+
f12	5.79E+04	4.39E+04	5.56E+04	1.35E+05	-	2.05E+05	7.38E+05	-	1.22E+06	1.33E+06	+	1.62E+05	4.04E+05	-	**1.97E+04**	1.56E+04	-	3.50E+04	5.08E+04	-	3.25E+06	2.55E+06	+	1.40E+07	8.60E+06	+
f13	**4.92E+03**	6.76E+03	1.10E+04	9.30E+03	+	8.34E+03	9.21E+03	=	9.73E+03	1.04E+04	+	1.09E+04	1.10E+04	+	7.51E+03	9.02E+03	=	1.15E+04	1.21E+04	+	1.11E+04	8.46E+03	+	9.27E+04	9.22E+04	+
f14	**8.09E+01**	3.66E+01	2.94E+03	5.00E+03	+	2.44E+03	4.61E+03	+	1.21E+03	3.49E+03	+	4.47E+03	6.59E+03	+	8.17E+03	8.55E+03	+	1.80E+03	3.62E+03	+	1.11E+02	6.09E+01	+	5.86E+02	8.38E+02	+
f15	**4.40E+02**	3.34E+02	4.41E+03	7.03E+03	+	2.75E+03	4.18E+03	+	2.98E+03	4.82E+03	+	6.85E+03	7.85E+03	+	6.11E+03	8.54E+03	+	6.37E+03	7.64E+03	+	1.01E+03	1.32E+03	=	2.79E+03	2.35E+03	+
f16	**5.18E+01**	6.35E+01	1.49E+02	1.02E+02	+	9.21E+01	8.74E+01	=	1.37E+02	1.02E+02	+	1.42E+02	1.12E+02	+	1.70E+02	1.39E+02	+	1.35E+02	1.19E+02	+	1.01E+02	6.29E+01	+	1.34E+02	8.92E+01	+
f17	**3.76E+01**	2.94E+01	4.44E+01	5.09E+01	=	4.62E+01	2.62E+01	+	4.02E+01	4.48E+01	=	4.29E+01	4.70E+01	=	5.13E+01	6.38E+01	=	1.09E+02	5.19E+01	+	7.20E+01	2.16E+01	+	1.13E+02	3.31E+01	+
f18	1.02E+04	9.11E+03	1.44E+04	1.34E+04	=	1.58E+04	1.25E+04	+	**9.55E+03**	8.31E+03	=	1.34E+04	1.14E+04	=	1.30E+04	1.22E+04	=	1.40E+04	9.36E+03	+	4.95E+04	5.64E+04	+	1.06E+05	1.17E+05	+
f19	**4.68E+02**	9.55E+02	5.20E+03	7.26E+03	+	8.77E+03	8.88E+03	+	5.13E+03	6.83E+03	+	5.36E+03	6.98E+03	+	1.04E+04	1.13E+04	+	1.01E+04	8.81E+03	+	1.05E+03	2.20E+03	+	5.51E+03	6.41E+03	+
f20	1.00E+01	1.07E+01	2.43E+00	4.09E+00	-	2.98E+01	1.53E+01	+	**1.26E+00**	1.77E+00	-	3.16E+00	4.64E+00	-	4.27E+00	6.90E+00	-	1.02E+02	5.17E+01	+	4.70E+01	1.28E+01	+	1.15E+02	2.56E+01	+
f21	1.67E+02	5.80E+01	1.80E+02	6.13E+01	+	1.42E+02	5.32E+01	=	1.89E+02	5.62E+01	+	1.92E+02	5.33E+01	+	2.16E+02	3.03E+01	+	1.35E+02	5.12E+01	=	**1.17E+02**	2.67E+01	=	1.21E+02	3.70E+01	=
f22	**8.66E+01**	3.13E+01	1.32E+02	1.46E+02	+	1.42E+02	3.98E+01	+	1.02E+02	1.04E+01	+	1.32E+02	1.65E+02	=	2.28E+02	2.96E+02	+	1.28E+02	7.87E+01	+	1.10E+02	1.92E+01	+	1.77E+02	4.31E+01	+
f23	**3.15E+02**	6.13E+00	3.20E+02	8.33E+00	+	3.23E+02	4.47E+01	+	3.22E+02	7.05E+00	+	3.21E+02	1.01E+01	+	3.19E+02	7.67E+00	+	3.21E+02	8.32E+00	+	3.50E+02	9.11E+00	+	3.40E+02	7.57E+00	+
f24	3.22E+02	6.89E+01	3.42E+02	6.33E+01	+	3.32E+02	6.42E+01	+	3.13E+02	1.03E+02	+	3.28E+02	8.77E+01	+	3.60E+02	5.23E+01	+	3.12E+02	7.57E+01	=	**2.41E+02**	1.05E+02	=	2.94E+02	9.93E+01	+
f25	**4.03E+02**	4.73E+01	4.28E+02	2.44E+01	+	4.61E+02	2.01E+01	+	4.23E+02	5.41E+01	+	4.24E+02	2.45E+01	+	4.32E+02	3.14E+01	+	4.58E+02	2.14E+01	+	4.46E+02	1.59E+01	+	4.74E+02	2.33E+01	+
f26	**3.20E+02**	1.91E+02	4.24E+02	2.66E+02	+	4.28E+02	7.09E+01	+	4.34E+02	3.44E+02	+	4.41E+02	3.30E+02	+	8.47E+02	5.45E+02	+	4.10E+02	7.14E+01	+	3.61E+02	4.66E+01	+	4.17E+02	3.44E+01	+
f27	**3.92E+02**	5.24E+00	4.13E+02	2.59E+01	+	4.01E+02	5.16E+00	+	4.05E+02	2.12E+01	+	4.05E+02	1.53E+01	+	4.06E+02	2.33E+01	+	3.98E+02	5.41E+00	+	4.11E+02	4.69E+00	+	3.99E+02	2.33E+00	+
f28	**3.70E+02**	1.15E+02	4.34E+02	1.43E+02	+	4.28E+02	4.21E+01	+	4.68E+02	1.30E+02	+	4.47E+02	1.37E+02	+	5.44E+02	1.44E+02	+	4.38E+02	8.28E+01	+	4.03E+02	4.88E+01	+	4.70E+02	7.87E+01	+
f29	**2.68E+02**	2.57E+01	3.04E+02	5.22E+01	+	2.88E+02	4.46E+01	+	3.18E+02	5.02E+01	+	3.08E+02	5.67E+01	+	3.20E+02	6.19E+01	+	3.36E+02	5.95E+01	+	3.36E+02	3.16E+01	+	3.39E+02	5.28E+01	+
f30	**1.26E+05**	3.51E+05	2.77E+05	4.14E+05	+	4.04E+05	5.98E+05	+	1.96E+05	3.03E+05	+	3.10E+05	4.54E+05	+	2.56E+05	4.19E+05	+	4.33E+05	5.86E+05	+	1.16E+06	1.39E+06	+	5.76E+05	5.90E+05	+

Table 20 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2017 in 10 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**	**Mean**	**Std**	**W**
f1	**2.19E+03**	2.39E+03	3.59E+03	2.93E+03	+	3.40E+03	2.94E+03	+	2.47E+03	2.28E+03	=
f2	1.94E+01	3.33E+01	**0.00E+00**	0.00E+00	-	1.35E+01	5.82E+00	+	5.90E+00	1.99E+01	=
f3	**1.38E-12**	9.10E-13	3.70E+03	5.45E+03	+	2.23E-10	4.16E-10	+	1.87E-12	7.16E-13	+
f4	6.32E-01	3.23E-01	1.33E+00	5.81E-01	+	2.01E+00	5.28E-01	+	**8.67E-03**	3.38E-02	-
f5	**1.46E+01**	4.84E+00	2.41E+02	1.17E+02	+	1.57E+01	6.57E+00	=	1.77E+01	7.23E+00	+
f6	**5.44E-04**	1.93E-03	8.26E+01	1.79E+01	+	1.04E+00	1.02E+00	+	6.26E-01	4.46E+00	-
f7	**2.35E+01**	7.14E+00	8.81E+02	2.58E+02	+	2.51E+01	8.05E+00	=	3.27E+01	9.15E+00	+
f8	1.57E+01	7.46E+00	1.48E+02	5.59E+01	+	**1.46E+01**	7.99E+00	=	2.45E+01	8.71E+00	+
f9	**3.56E-02**	2.42E-01	3.20E+03	1.38E+03	+	1.09E-01	4.00E-01	=	3.25E+02	4.18E+02	+
f10	5.92E+02	2.36E+02	1.86E+03	4.68E+02	+	**5.91E+02**	2.60E+02	=	6.58E+02	1.96E+02	=
f11	**1.32E+01**	6.38E+00	9.29E+01	5.92E+01	+	1.45E+01	6.98E+00	=	2.04E+01	1.19E+01	+
f12	5.79E+04	4.39E+04	1.28E+05	1.08E+05	+	5.20E+04	1.69E+04	=	**1.06E+04**	1.01E+04	-
f13	**4.92E+03**	6.76E+03	1.76E+04	1.29E+04	+	9.35E+03	1.03E+04	+	1.20E+04	1.06E+04	+
f14	8.09E+01	3.66E+01	6.31E+03	7.35E+03	+	**6.18E+01**	2.83E+01	-	7.15E+03	7.94E+03	+
f15	4.40E+02	3.34E+02	1.01E+04	1.09E+04	+	**3.85E+02**	4.73E+02	=	5.66E+03	6.96E+03	+
f16	5.18E+01	6.35E+01	7.83E+02	3.28E+02	+	**4.91E+01**	6.97E+01	=	4.85E+02	1.74E+02	+
f17	**3.76E+01**	2.94E+01	4.09E+02	2.26E+02	+	4.37E+01	2.51E+01	+	8.66E+01	1.03E+02	=
f18	**1.02E+04**	9.11E+03	1.30E+04	9.55E+03	=	2.54E+04	1.17E+04	+	1.20E+04	8.61E+03	=
f19	**4.68E+02**	9.55E+02	3.21E+03	5.25E+03	+	1.00E+03	1.10E+03	+	7.73E+03	9.65E+03	+
f20	1.00E+01	1.07E+01	5.76E+02	2.56E+02	+	3.70E+01	1.53E+01	+	**2.79E+00**	4.52E+00	-
f21	1.67E+02	5.80E+01	3.69E+02	1.09E+02	+	1.59E+02	6.28E+01	=	**1.01E+02**	1.31E+00	-
f22	**8.66E+01**	3.13E+01	1.73E+03	8.87E+02	+	8.88E+01	2.97E+01	=	9.97E+01	1.39E+01	=
f23	**3.15E+02**	6.13E+00	1.04E+03	4.84E+02	+	3.21E+02	9.90E+00	+	3.29E+02	4.94E+01	+
f24	3.22E+02	6.89E+01	4.76E+02	1.69E+02	+	3.42E+02	3.56E+01	=	**1.86E+02**	1.25E+02	-
f25	**4.03E+02**	4.73E+01	4.62E+02	1.18E+02	+	4.12E+02	2.10E+01	=	4.28E+02	3.45E+01	+
f26	**3.20E+02**	1.91E+02	2.26E+03	1.15E+03	+	3.46E+02	2.18E+02	-	3.90E+02	1.46E+02	+
f27	**3.92E+02**	5.24E+00	7.19E+02	3.55E+02	+	3.93E+02	2.50E+00	=	4.06E+02	1.01E+01	+
f28	**3.70E+02**	1.15E+02	6.10E+02	2.03E+02	+	3.91E+02	1.31E+02	=	4.02E+02	1.37E+02	=
f29	**2.68E+02**	2.57E+01	8.58E+02	2.32E+02	+	2.83E+02	4.38E+01	=	3.26E+02	5.57E+01	+
f30	**1.26E+05**	3.51E+05	7.99E+05	6.11E+05	+	4.38E+05	7.00E+05	+	3.97E+05	5.18E+05	+

Table 21 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2017 in 30 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	**2.30E+03**	3.01E+03	5.99E+03	5.74E+03	+	1.16E+09	1.14E+09	+	4.32E+04	3.49E+04	+	5.29E+03	6.23E+03	+	7.00E+03	6.93E+03	+	2.90E+09	4.24E+09	+	6.11E+09	1.24E+09	+	2.96E+10	5.03E+09	+
f2	6.62E+04	1.94E+05	1.45E+23	1.03E+24	+	1.42E+38	5.56E+38	+	2.47E+12	7.69E+12	+	6.78E+16	4.69E+17	+	**1.41E-03**	2.88E-03	-	1.63E+38	9.34E+38	+	2.45E+32	8.18E+32	+	1.07E+34	3.70E+34	+
f3	**9.89E-10**	1.17E-09	3.62E+04	1.93E+04	+	1.22E+05	3.09E+04	+	8.72E+03	5.96E+03	+	3.07E+04	1.65E+04	+	7.02E+03	7.77E+03	+	1.25E+05	3.16E+04	+	5.79E+04	9.80E+03	+	5.57E+04	1.73E+04	+
f4	8.07E+01	1.02E+01	1.13E+02	3.35E+01	+	2.09E+03	9.15E+02	+	8.91E+01	3.92E+01	=	7.27E+01	2.54E+01	-	**6.99E+01**	2.55E+01	-	3.70E+02	2.22E+02	+	1.54E+03	7.79E+02	+	7.07E+02	3.54E+02	+
f5	8.61E+01	2.30E+01	9.00E+01	2.56E+01	=	2.94E+02	4.01E+01	+	**8.32E+01**	2.32E+01	=	8.39E+01	2.35E+01	=	9.09E+01	2.04E+01	=	1.92E+02	4.78E+01	+	2.83E+02	2.44E+01	+	3.58E+02	2.84E+01	+
f6	4.13E-04	8.21E-04	4.81E-01	2.79E-01	+	4.75E+01	6.21E+00	+	6.62E-02	2.74E-02	+	4.91E-02	4.48E-02	+	**9.69E-07**	4.90E-07	-	3.98E+01	9.58E+00	+	3.88E+01	6.31E+00	+	7.11E+01	1.01E+01	+
f7	1.33E+02	2.61E+01	1.41E+02	3.16E+01	=	1.11E+03	1.30E+02	+	1.24E+02	2.53E+01	=	1.24E+02	2.51E+01	=	**1.05E+02**	1.96E+01	-	1.09E+03	2.42E+02	+	4.33E+02	4.31E+01	+	1.01E+03	1.31E+02	+
f8	1.01E+02	2.82E+01	1.01E+02	2.25E+01	=	2.83E+02	2.88E+01	+	**9.93E+01**	2.38E+01	=	1.00E+02	2.47E+01	=	1.11E+02	2.72E+01	+	2.16E+02	5.57E+01	+	2.68E+02	2.39E+01	+	3.36E+02	2.76E+01	+
f9	**9.32E+02**	8.65E+02	1.59E+03	1.15E+03	+	9.14E+03	2.38E+03	+	1.48E+03	1.10E+03	+	1.16E+03	1.23E+03	=	1.77E+03	1.78E+03	+	5.74E+03	1.82E+03	+	3.96E+03	1.30E+03	+	1.02E+04	2.33E+03	+
f10	2.85E+03	4.80E+02	2.97E+03	5.25E+02	=	5.70E+03	7.25E+02	+	2.75E+03	5.97E+02	=	**2.59E+03**	4.52E+02	-	2.69E+03	4.36E+02	=	4.67E+03	8.50E+02	+	7.09E+03	2.69E+02	+	7.50E+03	3.76E+02	+
f11	1.61E+02	5.33E+01	8.66E+01	3.44E+01	-	3.20E+03	1.83E+03	+	7.31E+01	3.63E+01	-	**5.44E+01**	3.00E+01	-	7.59E+01	3.64E+01	-	1.24E+03	1.19E+03	+	1.01E+03	3.48E+02	+	2.48E+03	9.63E+02	+
f12	8.87E+05	8.57E+05	7.49E+05	9.06E+05	-	1.42E+09	8.39E+08	+	3.44E+06	2.13E+06	+	1.09E+06	9.37E+05	=	**5.75E+05**	4.36E+05	-	5.65E+07	1.75E+08	+	2.93E+08	1.00E+08	+	6.03E+08	3.35E+08	+
f13	2.17E+04	2.35E+04	1.95E+04	2.01E+04	=	6.83E+07	1.69E+08	+	2.85E+04	9.92E+04	=	**1.86E+04**	2.08E+04	=	1.95E+04	2.01E+04	=	2.70E+06	1.54E+07	+	5.38E+07	3.35E+07	+	1.94E+08	1.46E+08	+
f14	**1.01E+04**	7.92E+03	7.96E+04	9.73E+04	+	3.43E+05	4.83E+05	+	8.81E+04	6.28E+04	+	3.83E+05	3.99E+05	+	1.18E+05	1.37E+05	+	1.10E+05	1.31E+05	+	6.78E+04	6.71E+04	+	2.23E+05	2.47E+05	+
f15	1.79E+04	1.39E+04	1.07E+04	1.00E+04	-	9.42E+04	2.25E+05	+	**9.00E+03**	8.38E+03	-	1.14E+04	1.03E+04	-	1.62E+04	1.49E+04	=	3.43E+04	2.02E+04	+	2.29E+06	2.51E+06	+	1.16E+07	1.23E+07	+
f16	**6.87E+02**	2.37E+02	1.15E+03	2.97E+02	+	1.68E+03	3.58E+02	+	1.02E+03	3.04E+02	+	1.15E+03	2.82E+02	+	1.27E+03	3.34E+02	+	1.10E+03	3.83E+02	+	2.19E+03	2.16E+02	+	2.06E+03	3.62E+02	+
f17	**2.83E+02**	1.70E+02	4.21E+02	2.18E+02	+	7.62E+02	2.38E+02	+	3.48E+02	1.58E+02	+	4.76E+02	2.04E+02	+	7.14E+02	2.54E+02	+	6.76E+02	2.52E+02	+	6.82E+02	1.83E+02	+	9.11E+02	2.04E+02	+
f18	**2.02E+05**	1.89E+05	6.17E+05	5.91E+05	+	5.76E+06	8.33E+06	+	2.33E+06	2.91E+06	+	1.67E+06	1.67E+06	+	5.49E+05	4.58E+05	+	8.71E+05	1.27E+06	+	2.41E+06	1.89E+06	+	3.65E+06	2.87E+06	+
f19	2.30E+04	2.07E+04	1.20E+04	1.07E+04	-	3.27E+06	5.10E+06	+	1.25E+04	1.20E+04	-	**1.01E+04**	1.15E+04	-	1.36E+04	1.22E+04	=	4.24E+05	1.31E+06	+	3.69E+06	2.85E+06	+	3.62E+07	2.38E+07	+
f20	**3.75E+02**	1.45E+02	5.75E+02	2.08E+02	+	5.91E+02	1.73E+02	+	4.54E+02	1.94E+02	+	5.75E+02	1.80E+02	+	5.84E+02	2.33E+02	+	5.96E+02	1.97E+02	+	5.85E+02	1.25E+02	+	7.91E+02	1.33E+02	+
f21	**2.86E+02**	2.26E+01	2.97E+02	2.43E+01	+	4.67E+02	4.17E+01	+	3.01E+02	2.49E+01	+	3.04E+02	2.62E+01	+	3.18E+02	3.44E+01	+	3.70E+02	3.97E+01	+	4.69E+02	2.17E+01	+	5.13E+02	3.35E+01	+
f22	2.61E+03	1.27E+03	2.99E+03	1.04E+03	=	3.78E+03	1.35E+03	+	2.93E+03	1.27E+03	=	3.03E+03	1.01E+03	=	2.59E+03	1.25E+03	=	1.96E+03	1.91E+03	=	**1.05E+03**	1.64E+02	-	3.13E+03	8.36E+02	=
f23	**4.35E+02**	2.61E+01	4.54E+02	2.67E+01	+	6.11E+02	4.21E+01	+	4.56E+02	3.28E+01	+	4.46E+02	2.33E+01	+	4.61E+02	2.72E+01	+	5.03E+02	4.06E+01	+	7.60E+02	5.37E+01	+	6.18E+02	2.86E+01	+
f24	**4.96E+02**	6.26E+01	6.08E+02	5.53E+01	+	6.54E+02	2.60E+01	+	6.05E+02	5.19E+01	+	6.32E+02	1.09E+02	+	7.04E+02	1.04E+02	+	5.44E+02	2.80E+01	+	8.32E+02	4.18E+01	+	6.69E+02	2.77E+01	+
f25	**3.86E+02**	1.21E+00	4.12E+02	2.55E+01	+	2.05E+03	6.48E+02	+	3.91E+02	1.04E+01	+	3.99E+02	1.69E+01	+	3.93E+02	1.61E+01	+	8.37E+02	3.20E+02	+	6.52E+02	7.98E+01	+	1.99E+03	4.32E+02	+
f26	1.95E+03	4.32E+02	2.13E+03	7.29E+02	+	4.10E+03	5.66E+02	+	**1.94E+03**	6.65E+02	=	2.12E+03	5.05E+02	+	2.28E+03	6.72E+02	+	2.92E+03	5.62E+02	+	4.59E+03	9.31E+02	+	4.08E+03	8.35E+02	+
f27	**5.17E+02**	9.77E+00	5.43E+02	1.64E+01	+	5.93E+02	2.67E+01	+	5.41E+02	1.78E+01	+	5.39E+02	1.76E+01	+	5.35E+02	1.98E+01	+	5.58E+02	3.34E+01	+	8.26E+02	8.64E+01	+	5.79E+02	2.69E+01	+
f28	**3.63E+02**	5.49E+01	4.49E+02	2.71E+01	+	1.56E+03	4.70E+02	+	4.23E+02	2.73E+01	+	3.97E+02	5.05E+01	+	3.79E+02	6.74E+01	=	8.86E+02	3.16E+02	+	1.04E+03	1.49E+02	+	1.17E+03	4.04E+02	+
f29	**7.81E+02**	1.72E+02	9.40E+02	2.62E+02	+	1.49E+03	2.82E+02	+	8.74E+02	2.50E+02	=	9.66E+02	2.49E+02	+	9.79E+02	2.32E+02	+	1.30E+03	2.80E+02	+	1.63E+03	2.52E+02	+	1.77E+03	3.12E+02	+
f30	2.26E+04	1.53E+04	1.04E+04	7.39E+03	-	1.10E+07	1.57E+07	+	1.11E+04	5.01E+03	-	**9.15E+03**	3.54E+03	-	1.11E+04	1.84E+04	-	2.55E+05	5.75E+05	+	1.35E+07	6.05E+06	+	8.82E+07	5.03E+07	+

Table 22 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2017 in 30 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	2.30E+03	3.01E+03	3.90E+03	3.97E+03	+	**2.09E+03**	1.40E+03	=	4.83E+03	5.80E+03	+
f2	6.62E+04	1.94E+05	**3.44E+03**	1.24E+04	-	1.28E+13	1.14E+13	+	4.37E+11	2.45E+12	=
f3	**9.89E-10**	1.17E-09	1.29E+05	6.09E+04	+	8.91E+02	2.26E+02	+	6.37E+04	3.90E+04	+
f4	8.07E+01	1.02E+01	5.99E+01	3.55E+01	-	7.52E+01	1.97E+01	-	**2.04E+01**	2.62E+01	-
f5	8.61E+01	2.30E+01	6.69E+02	1.66E+02	+	**8.31E+01**	2.00E+01	=	1.77E+02	4.98E+01	+
f6	**4.13E-04**	8.21E-04	9.29E+01	1.11E+01	+	1.80E+01	4.66E+00	+	6.92E-02	2.75E-01	-
f7	**1.33E+02**	2.61E+01	3.83E+03	7.76E+02	+	1.36E+02	2.94E+01	=	1.64E+02	3.72E+01	+
f8	1.01E+02	2.82E+01	5.62E+02	1.26E+02	+	**8.01E+01**	2.10E+01	-	1.86E+02	4.64E+01	+
f9	9.32E+02	8.65E+02	1.37E+04	3.83E+03	+	**7.04E+02**	6.61E+02	=	6.87E+03	2.52E+03	+
f10	**2.85E+03**	4.80E+02	5.66E+03	8.27E+02	+	3.69E+03	8.02E+02	+	3.36E+03	6.24E+02	+
f11	1.61E+02	5.33E+01	1.83E+02	5.93E+01	=	**9.67E+01**	3.76E+01	-	1.22E+02	4.60E+01	-
f12	8.87E+05	8.57E+05	1.52E+06	9.03E+05	+	8.07E+05	3.35E+05	=	**1.65E+05**	1.85E+05	-
f13	2.17E+04	2.35E+04	2.55E+04	2.31E+04	=	5.17E+04	2.79E+04	+	**1.99E+04**	2.11E+04	=
f14	**1.01E+04**	7.92E+03	1.76E+06	1.66E+06	+	1.04E+04	6.65E+03	=	9.51E+04	7.61E+04	+
f15	1.79E+04	1.39E+04	1.29E+04	1.33E+04	=	6.07E+04	1.73E+04	+	**1.24E+04**	1.14E+04	=
f16	**6.87E+02**	2.37E+02	1.85E+03	5.19E+02	+	8.09E+02	3.10E+02	+	1.41E+03	3.88E+02	+
f17	2.83E+02	1.70E+02	1.27E+03	4.03E+02	+	**2.77E+02**	1.43E+02	=	7.74E+02	2.57E+02	+
f18	2.02E+05	1.89E+05	5.42E+06	4.70E+06	+	**1.74E+05**	1.20E+05	=	5.08E+05	3.16E+05	+
f19	2.30E+04	2.07E+04	1.02E+04	1.33E+04	-	7.68E+04	2.67E+04	+	**9.89E+03**	1.15E+04	-
f20	3.75E+02	1.45E+02	1.35E+03	3.57E+02	+	**2.99E+02**	1.26E+02	-	8.10E+02	3.15E+02	+
f21	**2.86E+02**	2.26E+01	8.29E+02	1.85E+02	+	2.87E+02	2.30E+01	=	3.83E+02	4.19E+01	+
f22	**2.61E+03**	1.27E+03	6.25E+03	1.07E+03	+	3.73E+03	1.14E+03	+	3.97E+03	9.64E+02	+
f23	**4.35E+02**	2.61E+01	2.10E+03	7.93E+02	+	4.70E+02	3.13E+01	+	5.12E+02	6.66E+01	+
f24	**4.96E+02**	6.26E+01	1.15E+03	2.57E+02	+	5.48E+02	4.55E+01	+	7.73E+02	9.34E+01	+
f25	**3.86E+02**	1.21E+00	3.98E+02	1.59E+01	+	4.15E+02	1.67E+01	+	3.95E+02	1.69E+01	+
f26	**1.95E+03**	4.32E+02	9.32E+03	2.97E+03	+	2.06E+03	3.10E+02	=	3.02E+03	1.27E+03	+
f27	**5.17E+02**	9.77E+00	1.11E+03	7.52E+02	+	5.94E+02	1.82E+01	+	5.66E+02	2.75E+01	+
f28	3.63E+02	5.49E+01	3.62E+02	7.55E+01	=	3.66E+02	4.62E+01	=	**3.41E+02**	5.68E+01	-
f29	**7.81E+02**	1.72E+02	1.89E+03	3.71E+02	+	9.95E+02	2.16E+02	+	1.18E+03	3.09E+02	+
f30	2.26E+04	1.53E+04	**1.16E+04**	5.05E+03	-	4.29E+05	1.99E+05	+	1.28E+04	1.55E+04	-

Table 23 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2017 in 50 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	**5.99E+03**	6.65E+03	1.41E+07	3.06E+07	+	7.50E+09	7.63E+09	+	1.03E+05	7.25E+04	+	7.80E+03	9.38E+03	=	1.42E+04	1.30E+04	+	1.46E+10	9.13E+09	+	3.51E+10	4.92E+09	+	8.56E+10	1.17E+10	+
f2	**1.05E+13**	3.90E+13	8.31E+50	4.29E+51	+	4.93E+74	3.40E+75	+	6.71E+22	4.62E+23	+	1.36E+49	9.72E+49	+	1.49E+20	1.06E+21	-	1.74E+68	1.18E+69	+	2.47E+66	8.60E+66	+	2.57E+66	9.79E+66	+
f3	**1.76E-07**	1.72E-07	1.68E+05	3.26E+04	+	2.82E+05	4.45E+04	+	4.94E+04	1.85E+04	+	8.04E+04	2.35E+04	+	2.09E+04	1.27E+04	+	2.62E+05	4.27E+04	+	1.54E+05	2.00E+04	+	1.49E+05	2.73E+04	+
f4	1.13E+02	3.89E+01	2.74E+02	7.43E+01	+	1.20E+04	5.65E+03	+	1.46E+02	6.00E+01	+	1.02E+02	5.32E+01	=	**8.46E+01**	5.06E+01	-	2.11E+03	1.39E+03	+	7.58E+03	2.76E+03	+	8.64E+03	4.34E+03	+
f5	1.89E+02	4.15E+01	1.91E+02	3.82E+01	=	6.48E+02	5.11E+01	+	1.86E+02	3.35E+01	=	**1.84E+02**	4.14E+01	=	2.12E+02	3.76E+01	+	4.63E+02	8.46E+01	+	5.81E+02	3.02E+01	+	7.11E+02	4.68E+01	+
f6	1.75E-04	4.74E-04	3.17E+00	1.12E+00	+	7.23E+01	8.69E+00	+	8.90E-02	2.84E-02	+	4.49E-02	2.98E-02	+	**1.52E-06**	6.20E-07	-	5.48E+01	1.03E+01	+	5.88E+01	6.45E+00	+	9.14E+01	7.66E+00	+
f7	2.48E+02	4.20E+01	2.92E+02	3.91E+01	+	2.72E+03	2.37E+02	+	2.69E+02	4.14E+01	+	2.29E+02	3.89E+01	-	**2.04E+02**	2.93E+01	-	2.70E+03	5.20E+02	+	1.08E+03	9.28E+01	+	2.16E+03	2.96E+02	+
f8	1.85E+02	3.01E+01	2.14E+02	4.75E+01	+	6.52E+02	7.75E+01	+	1.94E+02	3.57E+01	=	**1.75E+02**	4.13E+01	=	2.11E+02	4.96E+01	+	4.41E+02	1.06E+02	+	5.71E+02	3.89E+01	+	7.27E+02	4.14E+01	+
f9	4.67E+03	2.57E+03	6.62E+03	3.22E+03	+	2.82E+04	6.76E+03	+	5.81E+03	2.62E+03	+	**3.18E+03**	2.30E+03	-	6.09E+03	4.05E+03	=	1.92E+04	6.79E+03	+	2.24E+04	5.34E+03	+	3.48E+04	5.08E+03	+
f10	5.05E+03	6.73E+02	5.20E+03	6.96E+02	=	1.13E+04	9.17E+02	+	4.79E+03	6.71E+02	=	**4.56E+03**	6.94E+02	-	4.85E+03	8.05E+02	=	8.64E+03	1.38E+03	+	1.33E+04	4.47E+02	+	1.36E+04	5.57E+02	+
f11	2.26E+02	5.19E+01	7.78E+02	1.34E+03	+	1.36E+04	5.85E+03	+	1.80E+02	5.39E+01	-	**1.37E+02**	3.61E+01	-	1.52E+02	4.33E+01	-	5.54E+03	5.05E+03	+	6.18E+03	1.64E+03	+	6.57E+03	2.90E+03	+
f12	3.89E+06	2.38E+06	1.48E+07	3.01E+07	+	1.38E+10	5.39E+09	+	8.85E+06	5.14E+06	+	2.61E+06	1.93E+06	-	**1.84E+06**	1.34E+06	-	1.25E+09	1.21E+09	+	5.06E+09	1.45E+09	+	7.53E+09	3.32E+09	+
f13	1.62E+04	1.37E+04	1.14E+04	9.25E+03	=	3.08E+09	2.26E+09	+	1.12E+04	9.56E+03	=	**9.27E+03**	9.67E+03	-	1.01E+04	1.02E+04	-	3.73E+08	6.45E+08	+	6.22E+08	3.57E+08	+	1.46E+09	6.17E+08	+
f14	**5.51E+04**	4.25E+04	2.28E+05	2.19E+05	+	2.62E+06	2.43E+06	+	6.04E+05	4.31E+05	+	6.11E+05	5.06E+05	+	1.92E+05	1.05E+05	+	1.22E+06	1.49E+06	+	1.86E+06	2.42E+06	+	1.84E+06	1.65E+06	+
f15	1.44E+04	9.47E+03	**7.02E+03**	7.57E+03	-	1.89E+08	2.51E+08	+	1.17E+04	1.73E+04	-	9.18E+03	6.09E+03	-	1.16E+04	9.03E+03	=	6.27E+05	4.12E+06	+	4.72E+07	2.16E+07	+	2.62E+08	2.10E+08	+
f16	**1.25E+03**	3.69E+02	2.11E+03	5.04E+02	+	3.60E+03	5.38E+02	+	1.88E+03	4.71E+02	+	2.13E+03	4.54E+02	+	2.13E+03	3.66E+02	+	2.16E+03	4.96E+02	+	4.23E+03	4.52E+02	+	4.24E+03	5.21E+02	+
f17	**1.16E+03**	2.87E+02	1.44E+03	3.08E+02	+	2.94E+03	5.62E+02	+	1.34E+03	3.60E+02	+	1.46E+03	2.67E+02	+	1.62E+03	3.14E+02	+	2.53E+03	5.58E+02	+	2.45E+03	2.19E+02	+	3.73E+03	4.80E+02	+
f18	**2.57E+05**	1.11E+05	2.62E+06	2.31E+06	+	3.03E+07	3.59E+07	+	3.38E+06	2.49E+06	+	1.52E+06	1.72E+06	+	6.88E+05	4.33E+05	+	7.00E+06	9.36E+06	+	1.46E+07	7.12E+06	+	1.28E+07	7.68E+06	+
f19	1.91E+04	1.71E+04	1.97E+04	1.24E+04	=	2.00E+08	3.03E+08	+	**1.61E+04**	1.37E+04	=	1.97E+04	1.42E+04	=	1.68E+04	1.53E+04	=	1.53E+06	7.38E+06	+	2.14E+07	9.57E+06	+	2.68E+08	1.39E+08	+
f20	**9.06E+02**	1.80E+02	1.12E+03	3.32E+02	+	1.72E+03	2.83E+02	+	1.13E+03	2.95E+02	+	1.11E+03	2.68E+02	+	1.31E+03	3.25E+02	+	1.49E+03	3.15E+02	+	1.89E+03	2.00E+02	+	1.96E+03	2.52E+02	+
f21	**3.82E+02**	4.00E+01	3.99E+02	3.86E+01	+	8.11E+02	7.08E+01	+	4.03E+02	5.24E+01	+	3.93E+02	4.50E+01	=	4.23E+02	4.67E+01	+	6.15E+02	6.75E+01	+	7.81E+02	4.11E+01	+	8.99E+02	4.18E+01	+
f22	5.89E+03	7.06E+02	5.96E+03	8.14E+02	=	1.14E+04	1.03E+03	+	5.74E+03	8.18E+02	=	5.61E+03	6.29E+02	-	**5.51E+03**	6.28E+02	-	9.08E+03	1.08E+03	+	1.37E+04	1.25E+03	+	1.35E+04	1.48E+03	+
f23	**6.13E+02**	3.65E+01	6.55E+02	5.17E+01	+	1.05E+03	6.69E+01	+	6.53E+02	4.78E+01	+	6.50E+02	5.01E+01	+	6.86E+02	4.67E+01	+	8.01E+02	7.23E+01	+	1.38E+03	8.36E+01	+	1.02E+03	5.80E+01	+
f24	**6.83E+02**	4.28E+01	9.27E+02	9.37E+01	+	1.03E+03	5.59E+01	+	9.32E+02	9.87E+01	+	9.60E+02	9.99E+01	+	1.18E+03	1.56E+02	+	7.98E+02	4.52E+01	+	1.40E+03	7.81E+01	+	1.04E+03	5.37E+01	+
f25	**5.07E+02**	3.34E+01	6.48E+02	5.03E+01	+	1.14E+04	3.15E+03	+	5.51E+02	3.80E+01	+	5.56E+02	3.89E+01	+	5.28E+02	6.24E+01	+	3.15E+03	2.17E+03	+	3.68E+03	1.02E+03	+	1.10E+04	3.03E+03	+
f26	**3.11E+03**	3.58E+02	3.55E+03	4.75E+02	+	7.87E+03	7.84E+02	+	3.50E+03	5.28E+02	+	3.46E+03	6.03E+02	+	3.73E+03	6.90E+02	+	5.08E+03	6.32E+02	+	1.07E+04	1.09E+03	+	7.36E+03	7.82E+02	+
f27	**6.19E+02**	4.77E+01	8.03E+02	1.15E+02	+	1.14E+03	1.25E+02	+	7.60E+02	8.78E+01	+	7.57E+02	9.32E+01	+	7.91E+02	1.09E+02	+	9.03E+02	1.08E+02	+	2.02E+03	2.49E+02	+	9.99E+02	1.30E+02	+
f28	**4.70E+02**	1.70E+01	6.34E+02	7.12E+01	+	6.74E+03	1.13E+03	+	5.02E+02	2.28E+01	+	4.97E+02	2.13E+01	+	4.93E+02	2.54E+01	+	2.98E+03	1.15E+03	+	3.97E+03	7.03E+02	+	4.55E+03	1.69E+03	+
f29	1.28E+03	2.58E+02	1.35E+03	3.13E+02	=	3.46E+03	7.33E+02	+	**1.17E+03**	2.99E+02	=	1.23E+03	3.24E+02	=	1.48E+03	2.63E+02	+	2.64E+03	6.09E+02	+	3.73E+03	4.92E+02	+	4.01E+03	5.87E+02	+
f30	2.06E+06	3.35E+06	1.24E+06	3.74E+05	=	6.56E+08	4.59E+08	+	1.18E+06	4.10E+05	=	**1.16E+06**	3.76E+05	=	1.24E+06	4.72E+05	=	7.09E+07	1.74E+08	+	2.12E+08	7.84E+07	+	5.48E+08	2.22E+08	+

Table 24 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2017 in 50 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	**5.99E+03**	6.65E+03	1.06E+04	1.12E+04	=	2.69E+07	7.94E+06	+	9.82E+03	9.56E+03	=
f2	**1.05E+13**	3.90E+13	2.31E+28	8.24E+28	=	2.92E+29	9.00E+29	+	1.21E+31	6.20E+31	=
f3	**1.76E-07**	1.72E-07	2.97E+05	7.93E+04	+	1.29E+04	1.51E+03	+	1.36E+05	4.15E+04	+
f4	1.13E+02	3.89E+01	7.84E+01	4.77E+01	-	1.88E+02	4.38E+01	+	**6.78E+01**	4.51E+01	-
f5	1.89E+02	4.15E+01	1.06E+03	2.01E+02	+	**1.74E+02**	3.37E+01	=	3.59E+02	8.27E+01	+
f6	**1.75E-04**	4.74E-04	9.23E+01	7.62E+00	+	3.06E+01	4.04E+00	+	6.92E-02	1.93E-01	=
f7	**2.48E+02**	4.20E+01	6.44E+03	1.08E+03	+	2.77E+02	4.68E+01	+	3.22E+02	6.23E+01	+
f8	1.85E+02	3.01E+01	9.93E+02	2.18E+02	+	**1.77E+02**	3.22E+01	=	3.84E+02	6.08E+01	+
f9	**4.67E+03**	2.57E+03	3.06E+04	5.95E+03	+	5.65E+03	5.88E+03	=	1.93E+04	4.80E+03	+
f10	**5.05E+03**	6.73E+02	8.97E+03	1.03E+03	+	7.03E+03	1.50E+03	+	6.06E+03	7.69E+02	+
f11	2.26E+02	5.19E+01	1.35E+03	1.79E+03	+	2.15E+02	2.76E+01	=	**1.81E+02**	5.01E+01	-
f12	3.89E+06	2.38E+06	2.10E+06	1.15E+06	-	6.78E+06	2.04E+06	+	**1.20E+06**	8.78E+05	-
f13	1.62E+04	1.37E+04	**1.40E+04**	1.03E+04	=	5.27E+04	2.19E+04	+	1.81E+04	1.46E+04	=
f14	5.51E+04	4.25E+04	2.09E+06	1.38E+06	+	**5.38E+04**	3.69E+04	=	2.33E+05	1.94E+05	+
f15	1.44E+04	9.47E+03	**9.96E+03**	7.84E+03	=	4.64E+04	1.22E+04	+	1.02E+04	7.54E+03	-
f16	**1.25E+03**	3.69E+02	2.91E+03	5.50E+02	+	1.45E+03	4.82E+02	=	2.23E+03	3.90E+02	+
f17	1.16E+03	2.87E+02	2.53E+03	4.62E+02	+	**1.08E+03**	3.01E+02	=	1.84E+03	4.23E+02	+
f18	**2.57E+05**	1.11E+05	2.27E+06	1.60E+06	+	2.65E+05	8.38E+04	=	1.05E+06	6.70E+05	+
f19	1.91E+04	1.71E+04	**1.16E+04**	1.31E+04	-	1.37E+05	4.08E+04	+	1.86E+04	1.48E+04	=
f20	9.06E+02	1.80E+02	2.56E+03	6.28E+02	+	**8.63E+02**	2.40E+02	=	1.58E+03	4.10E+02	+
f21	**3.82E+02**	4.00E+01	1.27E+03	1.83E+02	+	3.90E+02	3.19E+01	=	6.33E+02	8.56E+01	+
f22	**5.89E+03**	7.06E+02	9.40E+03	1.08E+03	+	8.00E+03	1.36E+03	+	6.93E+03	8.77E+02	+
f23	**6.13E+02**	3.65E+01	2.97E+03	8.65E+02	+	7.29E+02	5.42E+01	+	8.13E+02	1.81E+02	+
f24	**6.83E+02**	4.28E+01	1.72E+03	2.16E+02	+	8.37E+02	7.78E+01	+	1.34E+03	1.63E+02	+
f25	**5.07E+02**	3.34E+01	5.56E+02	3.20E+01	+	6.07E+02	3.39E+01	+	5.39E+02	4.19E+01	+
f26	**3.11E+03**	3.58E+02	1.50E+04	3.87E+03	+	4.12E+03	4.47E+02	+	5.41E+03	1.54E+03	+
f27	**6.19E+02**	4.77E+01	2.84E+03	1.39E+03	+	1.09E+03	8.94E+01	+	1.04E+03	2.65E+02	+
f28	**4.70E+02**	1.70E+01	5.04E+02	1.25E+01	+	6.65E+02	4.54E+01	+	5.00E+02	2.68E+01	+
f29	**1.28E+03**	2.58E+02	2.91E+03	4.35E+02	+	1.70E+03	4.36E+02	+	1.79E+03	4.51E+02	+
f30	2.06E+06	3.35E+06	**9.73E+05**	3.49E+05	-	3.87E+07	1.73E+06	+	1.04E+06	2.61E+05	-

Table 25 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against compact algorithms on CEC-2017 in 100 dimensions.

Function	cSM		cDE-Exp			McDE			DEcDE			CScDE			cSNUM			cFA			cPSO			cTLBO		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	**8.14E+03**	1.28E+04	5.02E+09	1.79E+09	+	3.79E+10	4.19E+10	+	6.35E+05	3.54E+05	+	1.18E+04	1.29E+04	+	2.24E+04	2.33E+04	+	9.49E+10	4.39E+10	+	1.75E+11	1.52E+10	+	2.93E+11	2.54E+10	+
f2	**2.32E+45**	1.54E+46	8.25E+135	5.89E+136	+	1.58E+169	Inf	+	4.66E+82	3.23E+83	+	5.57E+109	3.98E+110	+	1.29E+93	9.21E+93	-	1.33E+161	Inf	+	2.00E+156	Inf	+	1.70E+160	Inf	+
f3	**4.83E+03**	2.26E+03	6.05E+05	7.84E+04	+	7.51E+05	1.12E+05	+	2.68E+05	5.02E+04	+	3.01E+05	5.94E+04	+	1.85E+05	5.08E+04	+	6.81E+05	6.76E+04	+	4.50E+05	4.79E+04	+	3.54E+05	6.00E+04	+
f4	2.31E+02	2.72E+01	1.16E+03	2.97E+02	+	7.15E+04	1.42E+04	+	2.71E+02	4.58E+01	+	2.29E+02	3.67E+01	=	**2.05E+02**	3.85E+01	-	1.29E+04	6.70E+03	+	4.16E+04	1.05E+04	+	6.63E+04	1.55E+04	+
f5	**5.00E+02**	6.65E+01	7.13E+02	1.02E+02	+	1.66E+03	1.03E+02	+	5.45E+02	8.12E+01	+	5.02E+02	6.98E+01	=	6.16E+02	1.03E+02	+	1.21E+03	1.61E+02	+	1.47E+03	7.53E+01	+	1.73E+03	9.22E+01	+
f6	6.53E-05	1.06E-04	1.17E+01	1.66E+00	+	9.76E+01	8.46E+00	+	1.48E-01	4.98E-02	+	2.98E-02	1.26E-02	+	**3.23E-06**	1.18E-06	-	7.90E+01	1.04E+01	+	8.54E+01	5.15E+00	+	1.14E+02	6.71E+00	+
f7	7.07E+02	7.90E+01	1.25E+03	1.58E+02	+	7.94E+03	5.27E+02	+	8.47E+02	1.06E+02	+	6.46E+02	9.03E+01	-	**5.97E+02**	5.75E+01	-	7.75E+03	9.70E+02	+	3.98E+03	2.39E+02	+	5.13E+03	7.91E+02	+
f8	5.19E+02	7.77E+01	7.23E+02	9.29E+01	+	1.71E+03	1.21E+02	+	5.93E+02	8.78E+01	+	**5.06E+02**	7.70E+01	=	6.25E+02	1.09E+02	+	1.18E+03	1.75E+02	+	1.53E+03	7.91E+01	+	1.81E+03	8.62E+01	+
f9	1.84E+04	5.08E+03	3.73E+04	1.05E+04	+	9.25E+04	1.64E+04	+	2.80E+04	6.69E+03	+	**1.79E+04**	6.60E+03	=	2.58E+04	6.80E+03	+	5.71E+04	1.03E+04	+	8.15E+04	1.14E+04	+	1.11E+05	1.22E+04	+
f10	1.18E+04	1.00E+03	1.55E+04	1.25E+03	+	2.66E+04	1.24E+03	+	1.24E+04	1.09E+03	+	**1.17E+04**	1.21E+03	=	1.22E+04	1.11E+03	+	2.17E+04	2.20E+03	+	3.02E+04	5.04E+02	+	3.06E+04	6.74E+02	+
f11	1.29E+03	2.17E+02	5.63E+04	2.30E+04	+	2.19E+05	4.30E+04	+	4.86E+03	2.50E+03	+	1.41E+04	1.14E+04	+	**9.02E+02**	8.25E+02	-	1.21E+05	5.30E+04	+	1.17E+05	2.30E+04	+	9.70E+04	3.58E+04	+
f12	1.27E+07	4.80E+06	1.37E+09	7.52E+08	+	9.51E+10	1.94E+10	+	3.73E+07	1.44E+07	+	5.10E+06	3.18E+06	-	**2.37E+06**	8.59E+05	-	1.90E+10	1.05E+10	+	3.88E+10	6.61E+09	+	8.34E+10	2.53E+10	+
f13	1.29E+04	1.32E+04	3.84E+07	6.88E+07	+	1.39E+10	3.69E+09	+	1.09E+05	3.74E+05	+	**8.61E+03**	9.28E+03	=	9.62E+03	9.90E+03	=	1.39E+09	1.57E+09	+	3.58E+09	7.87E+08	+	1.02E+10	6.56E+09	+
f14	**1.67E+05**	7.05E+04	5.81E+06	5.80E+06	+	4.77E+07	2.79E+07	+	2.76E+06	1.33E+06	+	1.46E+06	8.52E+05	+	5.08E+05	2.73E+05	+	1.72E+07	1.90E+07	+	1.97E+07	9.03E+06	+	2.22E+07	8.45E+06	+
f15	8.26E+03	8.01E+03	1.69E+06	4.80E+06	+	3.62E+09	1.80E+09	+	1.60E+04	5.16E+04	=	5.99E+03	7.51E+03	-	**5.23E+03**	6.85E+03	-	1.87E+08	2.57E+08	+	5.60E+08	1.75E+08	+	2.02E+09	1.03E+09	+
f16	**3.39E+03**	6.43E+02	4.69E+03	7.41E+02	+	1.07E+04	1.15E+03	+	4.09E+03	7.15E+02	+	4.32E+03	7.17E+02	+	4.48E+03	6.73E+02	+	6.12E+03	1.00E+03	+	1.18E+04	1.20E+03	+	1.14E+04	1.17E+03	+
f17	**2.81E+03**	4.78E+02	3.69E+03	6.29E+02	+	5.52E+04	9.81E+04	+	3.29E+03	3.95E+02	+	3.16E+03	5.88E+02	+	3.60E+03	5.52E+02	+	6.47E+03	2.10E+03	+	7.64E+03	5.88E+02	+	1.06E+04	7.93E+03	+
f18	**3.66E+05**	1.29E+05	1.02E+07	7.79E+06	+	8.84E+07	5.79E+07	+	4.28E+06	2.08E+06	+	1.76E+06	8.71E+05	+	6.76E+05	3.16E+05	+	2.07E+07	1.83E+07	+	5.08E+07	2.20E+07	+	4.26E+07	1.99E+07	+
f19	1.17E+04	1.16E+04	1.20E+06	2.47E+06	+	4.79E+09	2.09E+09	+	1.19E+04	1.04E+04	=	**5.47E+03**	6.56E+03	-	8.39E+03	9.70E+03	=	3.92E+08	6.24E+08	+	5.93E+08	2.06E+08	+	1.71E+09	1.08E+09	+
f20	**2.66E+03**	4.39E+02	3.14E+03	6.65E+02	+	5.03E+03	4.74E+02	+	2.86E+03	4.13E+02	=	3.16E+03	6.00E+02	+	3.29E+03	6.08E+02	+	3.92E+03	6.58E+02	+	5.39E+03	2.16E+02	+	5.41E+03	2.59E+02	+
f21	**7.33E+02**	7.54E+01	9.33E+02	8.26E+01	+	1.92E+03	1.16E+02	+	8.18E+02	8.89E+01	+	7.42E+02	8.46E+01	=	8.80E+02	9.19E+01	+	1.40E+03	1.32E+02	+	1.80E+03	9.74E+01	+	2.00E+03	1.71E+02	+
f22	1.32E+04	1.24E+03	1.70E+04	1.38E+03	+	2.77E+04	1.26E+03	+	1.38E+04	1.18E+03	+	1.32E+04	8.63E+02	=	**1.30E+04**	1.27E+03	=	2.26E+04	2.02E+03	+	3.16E+04	4.31E+02	+	3.16E+04	8.13E+02	+
f23	9.18E+02	6.16E+01	1.01E+03	7.45E+01	+	1.95E+03	1.22E+02	+	8.81E+02	5.27E+01	-	**8.26E+02**	6.34E+01	-	8.95E+02	5.88E+01	=	1.45E+03	1.16E+02	+	2.77E+03	1.11E+02	+	2.02E+03	1.23E+02	+
f24	1.40E+03	7.62E+01	1.57E+03	8.69E+01	+	2.67E+03	1.52E+02	+	1.46E+03	1.04E+02	+	**1.37E+03**	7.21E+01	-	1.47E+03	8.18E+01	+	1.89E+03	1.49E+02	+	4.28E+03	3.08E+02	+	2.56E+03	1.71E+02	+
f25	**7.09E+02**	6.05E+01	1.97E+03	3.57E+02	+	4.56E+04	8.28E+03	+	8.19E+02	8.32E+01	+	7.61E+02	7.34E+01	+	7.47E+02	5.74E+01	+	1.65E+04	6.42E+03	+	1.66E+04	2.39E+03	+	3.84E+04	6.46E+03	+
f26	8.62E+03	8.60E+02	1.08E+04	8.81E+02	+	2.27E+04	1.85E+03	+	9.31E+03	8.33E+02	+	**8.53E+03**	7.50E+02	=	9.87E+03	1.01E+03	+	1.47E+04	1.54E+03	+	3.45E+04	2.63E+03	+	1.99E+04	2.02E+03	+
f27	**7.47E+02**	4.96E+01	9.70E+02	6.98E+01	+	2.33E+03	3.27E+02	+	8.44E+02	7.88E+01	+	8.64E+02	7.45E+01	+	8.71E+02	8.16E+01	+	1.27E+03	1.68E+02	+	4.88E+03	6.81E+02	+	1.47E+03	2.11E+02	+
f28	5.53E+02	3.30E+01	2.70E+03	1.16E+03	+	2.79E+04	3.48E+03	+	6.23E+02	4.07E+01	+	5.83E+02	3.22E+01	+	**5.50E+02**	3.23E+01	=	1.38E+04	2.64E+03	+	2.27E+04	2.75E+03	+	2.32E+04	6.37E+03	+
f29	3.95E+03	5.28E+02	3.89E+03	4.79E+02	=	2.95E+04	2.73E+04	+	**3.59E+03**	5.82E+02	-	3.62E+03	5.66E+02	-	3.78E+03	5.46E+02	=	9.75E+03	3.33E+03	+	1.00E+04	6.65E+02	+	2.63E+04	1.05E+04	+
f30	1.78E+04	6.67E+03	1.81E+07	2.39E+07	+	8.14E+09	2.71E+09	+	2.47E+04	8.55E+03	+	1.18E+04	6.72E+03	-	**9.01E+03**	5.37E+03	-	3.14E+08	4.99E+08	+	1.73E+09	4.02E+08	+	4.22E+09	1.85E+09	+

Table 26 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against single-solution algorithms on CEC-2017 in 100 dimensions.

Function	cSM		ISPO			nuSA			3SOME		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
f1	**8.14E+03**	1.28E+04	1.89E+04	1.94E+04	+	6.62E+09	5.31E+08	+	1.31E+04	1.59E+04	+
f2	2.32E+45	1.54E+46	**4.66E+37**	3.33E+38	-	2.01E+97	8.94E+97	+	4.49E+73	2.35E+74	=
f3	**4.83E+03**	2.26E+03	6.76E+05	1.24E+05	+	8.32E+04	4.18E+03	+	4.78E+05	9.74E+04	+
f4	2.31E+02	2.72E+01	**2.09E+02**	3.43E+01	-	9.96E+02	4.90E+01	+	2.12E+02	4.15E+01	=
f5	5.00E+02	6.65E+01	2.14E+03	2.70E+02	+	**4.82E+02**	3.96E+01	=	1.06E+03	1.14E+02	+
f6	**6.53E-05**	1.06E-04	9.40E+01	5.77E+00	+	4.36E+01	4.40E+00	+	2.01E-01	3.92E-01	+
f7	**7.07E+02**	7.90E+01	1.36E+04	1.48E+03	+	7.91E+02	1.01E+02	+	9.15E+02	1.30E+02	+
f8	5.19E+02	7.77E+01	2.30E+03	3.01E+02	+	**5.16E+02**	5.47E+01	=	1.11E+03	1.55E+02	+
f9	1.84E+04	5.08E+03	5.77E+04	5.77E+03	+	**1.61E+04**	5.90E+03	-	4.38E+04	7.59E+03	+
f10	**1.18E+04**	1.00E+03	1.78E+04	1.64E+03	+	1.76E+04	2.83E+03	+	1.43E+04	1.17E+03	+
f11	**1.29E+03**	2.17E+02	8.16E+04	4.04E+04	+	3.36E+03	2.35E+02	+	5.17E+03	5.02E+03	+
f12	1.27E+07	4.80E+06	7.21E+06	4.05E+06	-	3.64E+08	3.44E+07	+	**2.37E+06**	1.14E+06	-
f13	**1.29E+04**	1.32E+04	2.18E+04	1.34E+04	+	2.94E+04	5.12E+03	+	2.54E+04	1.46E+04	+
f14	**1.67E+05**	7.05E+04	1.92E+06	9.55E+05	+	1.89E+05	6.60E+04	=	1.47E+06	1.39E+06	+
f15	8.26E+03	8.01E+03	**5.80E+03**	5.00E+03	=	3.04E+04	8.12E+03	+	1.27E+04	9.74E+03	+
f16	**3.39E+03**	6.43E+02	5.45E+03	7.83E+02	+	4.19E+03	7.97E+02	+	4.37E+03	8.68E+02	+
f17	2.81E+03	4.78E+02	5.15E+03	8.00E+02	+	**2.64E+03**	4.65E+02	=	3.88E+03	6.36E+02	+
f18	3.66E+05	1.29E+05	1.80E+06	7.28E+05	+	**2.19E+05**	5.62E+04	-	1.03E+06	6.11E+05	+
f19	1.17E+04	1.16E+04	**5.96E+03**	6.28E+03	-	6.32E+05	1.53E+05	+	6.90E+03	7.89E+03	-
f20	**2.66E+03**	4.39E+02	5.24E+03	7.81E+02	+	2.67E+03	5.14E+02	=	3.85E+03	6.41E+02	+
f21	**7.33E+02**	7.54E+01	2.72E+03	3.42E+02	+	8.84E+02	7.67E+01	+	1.31E+03	1.54E+02	+
f22	**1.32E+04**	1.24E+03	2.00E+04	1.91E+03	+	1.98E+04	2.84E+03	+	1.56E+04	1.36E+03	+
f23	**9.18E+02**	6.16E+01	4.34E+03	1.13E+03	+	1.68E+03	1.04E+02	+	1.19E+03	1.13E+02	+
f24	**1.40E+03**	7.62E+01	4.26E+03	4.04E+02	+	2.49E+03	1.38E+02	+	1.89E+03	1.71E+02	+
f25	**7.09E+02**	6.05E+01	7.88E+02	7.05E+01	+	1.21E+03	3.67E+01	+	7.48E+02	5.45E+01	+
f26	**8.62E+03**	8.60E+02	3.49E+04	3.19E+03	+	1.47E+04	7.85E+02	+	1.50E+04	2.04E+03	+
f27	**7.47E+02**	4.96E+01	2.72E+03	2.07E+03	+	1.89E+03	1.31E+02	+	1.14E+03	1.74E+02	+
f28	5.53E+02	3.30E+01	**5.37E+02**	4.32E+01	-	1.61E+03	7.44E+01	+	5.67E+02	3.85E+01	=
f29	**3.95E+03**	5.28E+02	6.64E+03	8.99E+02	+	5.07E+03	4.59E+02	+	4.22E+03	5.78E+02	+
f30	1.78E+04	6.67E+03	1.73E+04	4.92E+03	=	2.76E+07	4.03E+06	+	**7.53E+03**	5.00E+03	-

Table 27 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against L-SHADE on CEC-2014 in 10, 30, 50 and 100 dimensions.

Function	*D* = 10					*D* = 30					*D* = 50					*D* = 100				
	cSM		**L-SHADE**			**cSM**		**L-SHADE**			**cSM**		**L-SHADE**			**cSM**		**L-SHADE**		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**
f1	1.01E+05	6.17E+04	**0.00E+00**	0.00E+00	-	1.02E+06	4.07E+05	**1.86E-04**	5.24E-04	-	1.82E+06	6.40E+05	**1.83E+04**	9.86E+03	-	5.51E+06	1.31E+06	**8.39E+05**	1.81E+05	-
f2	2.47E+03	2.66E+03	**0.00E+00**	0.00E+00	-	5.87E+03	8.37E+03	**3.07E-09**	6.72E-09	-	1.95E+03	3.10E+03	**2.49E-01**	2.29E-01	-	1.27E+04	1.63E+04	**2.47E+03**	1.48E+03	-
f3	1.18E+03	9.55E+02	**0.00E+00**	0.00E+00	-	1.05E+03	5.39E+02	**0.00E+00**	0.00E+00	-	3.56E+02	1.71E+02	**0.00E+00**	0.00E+00	-	1.29E+02	6.01E+01	**7.52E-04**	4.89E-04	-
f4	**9.25E+00**	1.51E+01	2.94E+01	1.26E+01	+	5.13E+01	4.03E+01	**4.79E+00**	9.03E+00	-	**9.06E+01**	1.90E+01	9.39E+01	4.91E+00	=	2.07E+02	3.12E+01	**1.92E+02**	2.49E+01	-
f5	2.00E+01	7.13E-05	**1.91E+01**	2.63E+00	+	**2.00E+01**	6.73E-06	2.07E+01	1.45E-01	+	**2.00E+01**	1.06E-03	2.10E+01	1.32E-01	+	**2.00E+01**	0.00E+00	2.13E+01	6.54E-02	+
f6	1.83E+00	1.34E+00	**5.60E-02**	2.09E-01	-	9.90E+00	3.53E+00	**4.97E-03**	7.61E-03	-	2.32E+01	4.93E+00	**2.66E-01**	5.23E-01	-	6.31E+01	7.99E+00	**8.70E+00**	2.33E+00	-
f7	1.50E-01	7.14E-02	**3.83E-02**	2.74E-02	-	1.12E-02	1.14E-02	**0.00E+00**	0.00E+00	-	4.01E-03	6.05E-03	**0.00E+00**	0.00E+00	-	2.03E-03	4.14E-03	**0.00E+00**	0.00E+00	-
f8	**2.12E-13**	8.51E-14	2.46E-02	5.64E-02	+	**6.11E-13**	1.34E-13	5.10E+01	3.66E+00	+	**1.95E-02**	1.39E-01	1.54E+02	8.29E+00	+	**5.85E-02**	2.36E-01	4.92E+02	1.68E+01	+
f9	1.28E+01	5.30E+00	**7.04E+00**	1.35E+00	-	**8.85E+01**	2.15E+01	9.66E+01	9.80E+00	+	**1.84E+02**	3.00E+01	2.21E+02	1.24E+01	+	**5.03E+02**	6.88E+01	5.81E+02	2.31E+01	+
f10	**1.43E+00**	2.08E+00	1.26E+01	3.91E+00	+	**1.64E+01**	3.21E+01	1.84E+03	2.26E+02	+	**2.50E+01**	4.23E+01	6.00E+03	3.03E+02	+	**6.68E+01**	7.62E+01	1.93E+04	4.96E+02	+
f11	5.42E+02	2.00E+02	**3.13E+02**	1.21E+02	-	**2.66E+03**	4.51E+02	4.80E+03	3.54E+02	+	**5.24E+03**	6.39E+02	1.15E+04	8.68E+02	+	**1.21E+04**	1.06E+03	2.92E+04	7.69E+02	+
f12	**7.39E-02**	6.68E-02	4.02E-01	6.56E-02	+	**1.24E-01**	5.28E-02	1.00E+00	1.44E-01	+	**1.20E-01**	3.14E-02	2.25E+00	6.68E-01	+	**2.16E-01**	4.30E-02	3.83E+00	4.03E-01	+
f13	1.85E-01	8.27E-02	**8.53E-02**	1.75E-02	-	4.56E-01	1.03E-01	**2.14E-01**	3.23E-02	-	6.08E-01	1.12E-01	**2.95E-01**	4.00E-02	-	6.46E-01	6.89E-02	**2.14E-01**	3.23E-02	-
f14	1.92E-01	7.61E-02	**1.10E-01**	3.02E-02	-	3.42E-01	2.08E-01	**2.95E-01**	2.88E-02	=	3.87E-01	2.41E-01	**3.68E-01**	2.79E-02	+	2.06E-01	1.81E-02	**1.79E-01**	2.02E-02	-
f15	1.12E+00	4.90E-01	**8.85E-01**	1.51E-01	-	**6.19E+00**	1.71E+00	9.93E+00	7.93E-01	+	**1.29E+01**	3.25E+00	2.31E+01	1.19E+00	+	**3.43E+01**	6.79E+00	5.85E+01	1.92E+00	+
f16	2.63E+00	3.80E-01	**2.16E+00**	2.51E-01	-	**1.10E+01**	5.57E-01	1.20E+01	4.16E-01	+	**1.95E+01**	8.14E-01	2.20E+01	3.95E-01	+	**4.18E+01**	9.08E-01	4.65E+01	2.54E-01	+
f17	6.32E+03	5.19E+03	**2.92E+01**	1.39E+01	-	1.37E+05	6.91E+04	**5.87E+02**	1.79E+02	-	1.84E+05	1.03E+05	**1.40E+03**	5.12E+02	-	5.83E+05	2.06E+05	**5.01E+03**	8.09E+02	-
f18	5.33E+03	8.52E+03	**1.58E+00**	6.43E-01	-	8.92E+03	7.04E+03	**3.10E+01**	4.84E+00	-	1.59E+03	1.27E+03	**9.75E+01**	1.38E+01	-	2.66E+03	2.11E+03	**4.32E+02**	2.83E+01	-
f19	1.48E+00	6.84E-01	**8.66E-01**	2.74E-01	-	9.35E+00	1.90E+00	**4.82E+00**	3.88E-01	-	2.15E+01	6.12E+00	**1.22E+01**	3.72E-01	-	1.07E+02	1.68E+01	**9.69E+01**	2.02E+00	-
f20	1.18E+03	3.31E+03	**1.39E+00**	3.48E-01	-	5.90E+02	3.42E+02	**1.94E+01**	2.77E+00	-	7.86E+02	2.38E+02	**6.74E+01**	8.74E+00	-	1.18E+03	2.22E+02	**2.10E+02**	5.08E+01	-
f21	2.20E+03	2.03E+03	**2.24E+00**	1.13E+00	-	6.63E+04	3.94E+04	**4.31E+02**	1.33E+02	-	2.13E+05	9.71E+04	**8.02E+02**	2.67E+02	-	3.54E+05	1.49E+05	**2.68E+03**	5.22E+02	-
f22	2.43E+01	1.13E+01	**1.10E+01**	2.98E+00	-	3.80E+02	1.61E+02	**2.08E+02**	7.46E+01	-	**8.05E+02**	2.85E+02	8.10E+02	1.44E+02	=	**1.90E+03**	5.15E+02	3.14E+03	2.46E+02	+
f23	**3.23E+02**	4.61E+01	3.29E+02	2.87E-13	-	3.15E+02	9.02E-04	**3.15E+02**	4.02E-13	-	3.44E+02	2.47E-01	**3.44E+02**	4.44E-13	-	3.51E+02	2.01E+00	**3.48E+02**	7.56E-07	-
f24	1.25E+02	7.64E+00	**1.12E+02**	1.90E+00	-	2.27E+02	1.04E+01	**2.24E+02**	1.05E+00	-	**2.70E+02**	5.14E+00	2.75E+02	6.62E-01	+	**3.57E+02**	8.26E+00	3.94E+02	2.87E+00	+
f25	1.88E+02	2.69E+01	**1.41E+02**	3.58E+01	-	2.06E+02	1.32E+00	**2.03E+02**	4.96E-02	-	2.13E+02	2.71E+00	**2.05E+02**	3.65E-01	-	2.42E+02	5.63E+00	**2.00E+02**	3.42E-07	-
f26	1.00E+02	7.36E-02	**1.00E+02**	2.01E-02	-	1.00E+02	1.02E-01	**1.00E+02**	2.84E-02	-	1.07E+02	2.38E+01	**1.02E+02**	1.40E+01	-	**1.56E+02**	5.10E+01	2.00E+02	1.19E-02	=
f27	2.12E+02	1.99E+02	**5.84E+01**	1.34E+02	-	5.00E+02	1.06E+02	**3.00E+02**	2.74E-04	-	8.87E+02	1.33E+02	**3.33E+02**	3.03E+01	-	1.81E+03	2.68E+02	**3.77E+02**	3.28E+01	-
f28	4.13E+02	5.55E+01	**3.81E+02**	3.17E+01	-	9.46E+02	8.24E+01	**8.40E+02**	1.40E+01	-	1.66E+03	3.26E+02	**1.11E+03**	2.91E+01	-	4.80E+03	1.12E+03	**2.31E+03**	4.56E+01	-
f29	7.83E+02	5.90E+02	**2.22E+02**	4.63E-01	-	5.89E+03	6.28E+03	**7.18E+02**	5.15E+00	-	1.19E+04	1.63E+04	**8.36E+02**	3.93E+01	-	9.53E+03	1.32E+04	**1.96E+03**	2.28E+02	-
f30	7.70E+02	1.92E+02	**4.65E+02**	1.34E+01	-	4.47E+03	1.75E+03	**1.30E+03**	5.83E+02	-	1.57E+04	4.39E+03	**8.67E+03**	4.13E+02	-	3.57E+04	1.54E+04	**8.59E+03**	1.04E+03	-
-/=/+				25/0/5					21/1/8					18/2/10					19/1/10	

Table 28 Average error ± standard deviation and Wilcoxon rank-sum test (reference: cSM) for cSM against JSO on CEC-2017 in 10, 30, 50 and 100 dimensions.

Function	$D = 10$					$D = 30$					$D = 50$					$D = 100$				
	cSM		**JSO**			**cSM**		**JSO**			**cSM**		**JSO**			**cSM**		**JSO**		
	Mean	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**	**Mean**	**Std**	**Mean**	**Std**	**W**
f1	2.19E+03	2.39E+03	**0.00E+00**	0.00E+00	-	2.30E+03	3.01E+03	**3.68E-06**	1.95E-06	-	5.99E+03	6.65E+03	**2.44E-01**	2.97E-01	-	8.14E+03	1.28E+04	**3.92E+02**	2.47E+02	-
f2	1.94E+01	3.33E+01	**0.00E+00**	0.00E+00	-	6.62E+04	1.94E+05	**1.72E-08**	6.48E-08	-	1.05E+13	3.90E+13	**3.77E+04**	1.96E+05	-	2.32E+45	1.54E+46	**6.81E+30**	4.53E+31	-
f3	1.38E-12	9.10E-13	**0.00E+00**	0.00E+00	-	**9.89E-10**	1.17E-09	4.03E-08	1.98E-08	+	**1.76E-07**	1.72E-07	2.18E-04	1.56E-04	+	4.83E+03	2.26E+03	**5.20E+01**	3.77E+01	-
f4	6.32E-01	3.23E-01	**0.00E+00**	0.00E+00	-	8.07E+01	1.02E+01	**5.87E+01**	7.78E-01	-	1.13E+02	3.89E+01	**8.19E+01**	5.67E+01	-	2.31E+02	2.72E+01	**2.11E+02**	1.38E+01	-
f5	1.46E+01	4.84E+00	**9.31E+00**	2.39E+00	-	**8.61E+01**	2.30E+01	1.25E+02	1.07E+01	+	**1.89E+02**	4.15E+01	2.72E+02	1.16E+01	+	**5.00E+02**	6.65E+01	6.52E+02	2.15E+01	+
f6	5.44E-04	1.93E-03	**3.57E-05**	1.53E-05	+	4.13E-04	8.21E-04	**2.92E-04**	5.66E-05	+	1.75E-04	4.74E-04	**9.82E-05**	2.94E-05	+	**6.53E-05**	1.06E-04	2.78E-04	6.93E-04	+
f7	2.35E+01	7.14E+00	**2.27E+01**	2.89E+00	=	**1.33E+02**	2.61E+01	1.70E+02	1.33E+01	+	**2.48E+02**	4.20E+01	3.39E+02	1.56E+01	+	**7.07E+02**	7.90E+01	7.87E+02	1.97E+01	+
f8	1.57E+01	7.46E+00	**1.04E+01**	2.56E+00	-	**1.01E+02**	2.82E+01	1.25E+02	1.23E+01	+	**1.85E+02**	3.01E+01	2.71E+02	1.76E+01	+	**5.19E+02**	7.77E+01	6.50E+02	1.76E+01	+
f9	3.56E-02	2.42E-01	**0.00E+00**	0.00E+00	-	9.32E+02	8.65E+02	**0.00E+00**	0.00E+00	-	4.67E+03	2.57E+03	**0.00E+00**	0.00E+00	-	1.84E+04	5.08E+03	**4.59E-02**	1.15E-01	-
f10	**5.92E+02**	2.36E+02	7.01E+02	1.78E+02	+	**2.85E+03**	4.80E+02	6.19E+03	3.00E+02	+	**5.05E+03**	6.73E+02	1.21E+04	3.37E+02	+	**1.18E+04**	1.00E+03	2.82E+04	5.79E+02	+
f11	1.32E+01	6.38E+00	**3.22E+00**	5.52E-01	-	1.61E+02	5.33E+01	**3.19E+01**	1.35E+01	-	2.26E+02	5.19E+01	**9.67E+01**	8.37E+00	-	1.29E+03	2.17E+02	**2.44E+02**	7.60E+01	-
f12	5.79E+04	4.39E+04	**2.44E+01**	1.96E+01	-	8.87E+05	8.57E+05	**5.78E+02**	3.04E+02	-	3.89E+06	2.38E+06	**1.75E+03**	5.18E+02	-	1.27E+07	4.80E+06	**5.55E+04**	2.36E+04	-
f13	4.92E+03	6.76E+03	**7.74E+00**	1.88E+00	-	2.17E+04	2.35E+04	**6.68E+01**	8.38E+00	-	1.62E+04	1.37E+04	**1.94E+02**	3.40E+01	-	1.29E+04	1.32E+04	**6.12E+02**	7.05E+01	-
f14	8.09E+01	3.66E+01	**7.18E+00**	1.89E+00	-	1.01E+04	7.92E+03	**4.86E+01**	5.59E+00	-	5.51E+04	4.25E+04	**1.01E+02**	7.72E+00	-	1.67E+05	7.05E+04	**2.88E+02**	1.69E+01	-
f15	4.40E+02	3.34E+02	**6.38E-01**	1.94E-01	-	1.79E+04	1.39E+04	**2.36E+01**	3.04E+00	-	1.44E+04	9.47E+03	**8.36E+01**	8.41E+00	-	8.26E+03	8.01E+03	**2.24E+02**	6.01E+01	-
f16	5.18E+01	6.35E+01	**5.41E+00**	1.44E+00	-	**6.87E+02**	2.37E+02	8.20E+02	1.79E+02	+	**1.25E+03**	3.69E+02	1.96E+03	1.98E+02	+	**3.39E+03**	6.43E+02	6.08E+03	3.31E+02	+
f17	3.76E+01	2.94E+01	**3.44E+01**	3.72E+00	+	2.83E+02	1.70E+02	**2.14E+02**	2.57E+01	=	**1.16E+03**	2.87E+02	1.32E+03	1.29E+02	+	**2.81E+03**	4.78E+02	4.02E+03	2.21E+02	+
f18	1.02E+04	9.11E+03	**4.84E-01**	2.04E-01	-	2.02E+05	1.89E+05	**3.05E+01**	1.84E+00	-	2.57E+05	1.11E+05	**5.39E+01**	5.61E+00	-	3.66E+05	1.29E+05	**1.80E+02**	3.52E+01	-
f19	4.68E+02	9.55E+02	**1.19E+00**	3.34E-01	-	2.30E+04	2.07E+04	**2.41E+01**	1.70E+00	-	1.91E+04	1.71E+04	**5.07E+01**	4.64E+00	-	1.17E+04	1.16E+04	**1.26E+02**	2.18E+01	-
f20	**1.00E+01**	1.07E+01	2.15E+01	3.97E+00	+	3.75E+02	1.45E+02	**3.12E+02**	4.74E+01	-	**9.06E+02**	1.80E+02	1.16E+03	1.22E+02	+	**2.66E+03**	4.39E+02	4.33E+03	2.00E+02	+
f21	1.67E+02	5.80E+01	**1.40E+02**	5.13E+01	-	**2.86E+02**	2.26E+01	3.19E+02	9.95E+00	+	**3.82E+02**	4.00E+01	4.73E+02	1.16E+01	+	**7.33E+02**	7.54E+01	8.78E+02	2.40E+01	+
f22	**8.66E+01**	3.13E+01	1.00E+02	1.80E-01	-	2.61E+03	1.27E+03	**1.00E+02**	0.00E+00	-	**5.89E+03**	7.06E+02	7.55E+03	5.81E+03	+	**1.32E+04**	1.24E+03	2.92E+04	6.49E+02	+
f23	3.15E+02	6.13E+00	**3.07E+02**	2.15E+00	-	**4.35E+02**	2.61E+01	4.67E+02	1.08E+01	+	**6.13E+02**	3.65E+01	6.89E+02	1.78E+01	+	**9.18E+02**	6.16E+01	1.18E+03	1.89E+01	+
f24	**3.22E+02**	6.89E+01	3.28E+02	3.57E+01	-	**4.96E+02**	6.26E+01	5.31E+02	8.43E+00	+	**6.83E+02**	4.28E+01	7.43E+02	1.87E+01	+	**1.40E+03**	7.62E+01	1.48E+03	3.07E+01	+
f25	**4.03E+02**	4.73E+01	4.06E+02	1.75E+01	-	**3.86E+02**	1.21E+00	3.87E+02	7.72E-03	-	5.07E+02	3.34E+01	**4.81E+02**	2.80E+00	-	**7.09E+02**	6.05E+01	7.36E+02	3.52E+01	+
f26	3.20E+02	1.91E+02	**3.00E+02**	0.00E+00	=	**1.95E+03**	4.32E+02	2.00E+03	9.68E+01	=	**3.11E+03**	3.58E+02	3.44E+03	1.71E+02	+	**8.62E+03**	8.60E+02	8.83E+03	5.89E+02	+
f27	3.92E+02	5.24E+00	**3.89E+02**	2.22E-01	-	5.17E+02	9.77E+00	**4.99E+02**	7.99E+00	-	6.19E+02	4.77E+01	**5.11E+02**	1.11E+01	-	7.47E+02	4.96E+01	**5.86E+02**	2.16E+01	-
f28	3.70E+02	1.15E+02	**3.43E+02**	1.00E+02	=	3.63E+02	5.49E+01	**3.13E+02**	3.36E+01	-	4.70E+02	1.70E+01	**4.60E+02**	6.84E+00	-	5.53E+02	3.30E+01	**5.40E+02**	2.41E+01	-
f29	2.68E+02	2.57E+01	**2.57E+02**	6.87E+00	-	7.81E+02	1.72E+02	**7.51E+02**	3.73E+01	=	1.28E+03	2.58E+02	**1.13E+03**	1.15E+02	-	**3.95E+03**	5.28E+02	4.26E+03	2.50E+02	+
f30	1.26E+05	3.51E+05	**3.95E+02**	1.01E-01	-	2.26E+04	1.53E+04	**1.98E+03**	1.88E+01	-	2.06E+06	3.35E+06	**6.01E+05**	2.98E+04	-	1.78E+04	6.67E+03	**2.47E+03**	1.46E+02	-
-/=/+				23/3/4					17/3/10					16/0/14					15/0/15	

Table 29 List of symbols used in the paper. Symbols written in boldface indicate vectors/matrices.

Symbol	Description
D	Problem size (no. variables)
B	Non-uniform mutation parameter (used in NUM)
ρ	Uniform random number in [0,1] (used in NUM)
η	Random digit in {-1,1} (used in NUM)
T	Total budget (no. evaluations)
i,k	Variable index
t	Iteration index
λ	Initialization constant
x_k	k-th element of solution $\boldsymbol{x}$
L_k	Lower bound for k-th variable
U_k	Upper bound for k-th variable
N_P	Virtual population size
$budget_1$	Budget for Module-1 (no. evaluations)
$budget_2$	Budget for Module-1 (no. evaluations)
$\boldsymbol{x}$	Parent solution (D-dimensional vector)
$\boldsymbol{x_{off}}$	Offspring solution (D-dimensional vector)
$\boldsymbol{lbest}$	Local best for Module-2 D-dimensional vector)
$\boldsymbol{elite}$	Global best solution of the algorithm D-dimensional vector)
$\boldsymbol{winner}$	Winner solution (D-dimensional vector)
$\boldsymbol{loser}$	Loser solution (D-dimensional vector)
$\boldsymbol{PV}$	Probability vector ($2 \times D$-dimensional matrix)
$\boldsymbol{\mu}$	Mean vector (D-dimensional vector)
$\boldsymbol{\sigma}$	Standard deviation vector (D-dimensional vector)