A Compact Compound Sinusoidal Differential Evolution Algorithm for Solving Optimisation Problems in Memory-Constrained Environments

Souheila Khalfi[a], Amer Draa[b,*], Giovanni Iacca[c]

[a]*Department of Fundamental Informatics and its Applications, Constantine 2 University, Algeria*
[b]*Department of Fundamental Informatics and its Applications, Constantine 2 University, Algeria*
[c]*Department of Information Engineering and Computer Science, University of Trento, Italy*

Abstract

In this paper, a new compact algorithm is proposed. Two sinusoidal formulas are used to automatically adjust the crossover rate and the mutation scaling factor in the compact Differential Evolution (cDE) metaheuristic. The proposed algorithm, called Compound Sinusoidal cDE, CScDE, is compared to seven state-of-the-art compact algorithms on the well-known BBOB test-bed, the CEC-2014 test suite for continuous optimisation, as well as five real-world optimisation problems chosen from the CEC-2011 benchmarks. The CScDE algorithm outperformed its competitors for most problem categories and over most dimensions. It is also compared to some well-established population-based metaheuristics.

Keywords: Limited-Memory Hardware, Compact Optimisation, Evolutionary Algorithms, Parameter Setting, Compound Sinusoidal Differential Evolution.

1. Introduction

In the last decade, the field of Artificial Intelligence (AI) has known exponential evolution in some areas, such as machine and deep learning. This has made it possible to process huge amounts of data to extract information and take relatively-autonomous decisions. Besides, it has become possible to endow some hardware-limited devices with algorithms that process data in a much faster and accurate way. Autonomous vehicle driving, satellite/aerial image segmentation and enhancement, and the internet of things are examples of applications in which expert and intelligent systems have been successfully implemented in industry. In many situations, these systems require solving some optimisation problems using the possibilities offered by intelligent approaches, such as metaheuristics, most of which are known to belong to the class of Evolutionary Algorithms (EA). A large set of EAs are known to be population-based, they operate by evolving a population of solutions over a given number of generations. It is relatively true that computational resources in terms of memory space are no longer an issue in modern computers. Though exploiting population-based metaheuristics in some specific applications, such as embedded systems and other similar environments, which have cost and/or space limitations, is usually a challenge. A population of individuals of a given EA is likely not to fit into the allocated memory that can be comfortably afforded.

*Corresponding author
Email addresses: souheila.khalfi@univ-constantine2.dz (Souheila Khalfi), amer.draa@univ-constantine2.dz (Amer Draa), giovanni.iacca@unitn.it (Giovanni Iacca)

To respond to this need, the metaheuristics community has thought of some alternative solutions in which memory limitations are explicitly considering when devising the optimisation algorithm. This has given birth to what is known as compact algorithms/optimisers. Thanks to their low memory requirements, compact algorithms have been widely accepted in recent years as an option to treat optimisation problems. They refer to the set of algorithms using the same logic of search of traditional population-based metaheuristics, but without any need to process or store an entire population of solutions (Neri et al., 2013a). In the hope of achieving a trade-off between global and local search forces; the optimisation problem-solving process tends to focus on global exploration at the beginning stages, but by laps of time, the algorithm would concentrate on the local exploration of the already-found solutions.

Compact metaheuristics could be seen as a sub-class of estimation of distribution algorithms, which are based on the idea of using one representative of solutions instead of a whole population, with contrast to what is used in the case of population-based algorithms. These optimisers are particularly simple, computationally-lightweight but very efficient. The most important benefit offered here is that the adopted probabilistic model needs much smaller memory space requirements to be run, compared to the population-based counterparts. These features are what make these algorithms very interesting to use in some specific implementations of intelligent systems. Consequently, a totally new domain known as *compact optimisation* (Neri et al., 2013a) has emerged in the last decade, and is even getting more attention in recent years. Although the mentioned advantages of compact optimisation algorithms, two disadvantages have been observed (Iacca & Caraffini, 2020). The first drawback is the premature convergence resulting from diversity loss in later generations of the search process. This is mainly caused by the absence of an entire population –replaced by an estimated probabilistic model. The existence of many competing/cooperating candidate solution is likely to boost the exploration efficiency of the algorithm. It is a fact that this is an important contributor to the success of population-based algorithms (Prügel-Bennett, 2010). The second limitation comes from the fact that compact algorithms do not consider any potential dependencies among the variables of the optimised solution. Unfortunately, most optimisation problems are non-separable by nature. In effect, in some complex problems, in which variables interact with each other, compact optimisers may fail.

The Differential Evolution (DE) algorithm, first proposed in (Storn & Price, 1997), is one of the most efficient continuous optimisation metaheuristics. Over the last couple of decades, this optimiser has become extremely popular and has been successfully applied to tackle both academic and real-world problems. Being simple yet powerful, the DE has undergone many improvements, which resulted in many variants proposed in the literature, mainly for boosting its performances or adapting it to handle complex problems. There are three control parameters involved in the DE standard variety: the population size, the mutation scaling factor, and the crossover rate. As any other evolutionary algorithm, setting these DE parameters is likely to influence the algorithm's overall performance. Their impact on the search process closely depends on the properties of the problem being targeted. Multiple attempts have been conducted to improve the DE by: adapting its parameters (Draa et al., 2015, 2018), developing new mutation strategies and crossover forms (Tian & Gao, 2019; Prabha & Yadav, 2020), preserving population diversity (Zhu et al., 2013; Yang et al., 2014a) or even by using some search mechanisms taken from other evolutionary/swarm intelligence approaches (dos Santos Coelho et al., 2009; Yildizdan & Baykan, 2020). In an analogous manner, many publications attempted to improve the behaviour of one particular compact algorithm: *the compact cDE*. This has led to a wide range of its variants: the Supervised compact Differential

Evolution (ScDE) (Iacca et al., 2011a), Composed compact Differential Evolution (CcDE) (Iacca et al., 2011b), Compact differential evolution light (cDElight) (Iacca et al., 2012a), Disturbed Exploitation compact Differential Evolution DEcDE (Neri et al., 2011), Memetic compact Differential Evolution (McDE) (Neri & Mininno, 2010), Opposition-Based Learning in compact Differential Evolution (Iacca et al., 2011c), Covariance Matrix compact Differential Evolution (CMcDE)(Jewajinda, 2016) and, more recently, cDE-light with Re-Sampled Inheritance (RI-cDE) (Iacca & Caraffini, 2020). Although the existence of a large number of works on cDE variants, there is still much room for improving cDE performances. Hence, it warrants special consideration.

Motivated by the good results offered by the Sinusoidal DE, proposed in (Draa et al., 2015), and its OCSinDE extension (Draa et al., 2018), we propose in this paper a new compact DE variant: we call it CScDE, for Compound Sinusoidal cDE. This algorithm has the advantages as other compact metaheuristics in terms of limited-memory needs, on the one hand, and those of the OCSinDE metaheuristic in terms of simplicity of the formulas used for adjusting parameter values of the DE algorithm, on the other hand.

The rest of the paper is organised as follows. Section 2 presents the basic concepts and the main issues related to compact metaheuristics. Section 3 is devoted to the explanation of the proposed CScDE. The experimental study and detailed comparisons of results are given in Section 4. Finally, Section 5 concludes the paper.

2. Compact optimisation

Historically speaking, the compact Genetic Algorithm (cGA) (Harik et al., 1999) was the first to be developed and introduced in the literature. Many extensions have been later proposed, where the main objective has been to improve performances. Subsequently, this "compact" logic has been extended to other metaheuristics. We can distinguish the following groups of methods:

- Compact variants of Evolutionary Algorithms (cEAs): the compact Genetic Algorithm with real encoding (rcGA) (Mininno et al., 2008), compact Differential Evolution (cDE) (Mininno et al., 2011), compact Evolutionary Strategies (Sergio et al., 2014) and Covariance Matrix compact Differential Evolution (CMcDE)(Jewajinda, 2016);

- Compact algorithms of Swarm Intelligence (cSIs): compact Bacterial Foraging Optimisation (cBFO) (Iacca et al., 2012b), compact Particle Swarm Optimisation (cPSO) (Neri et al., 2013b), compact Artificial Bee Colony (cABC)(Dao et al., 2014a) (Banitalebi et al., 2015), Enhanced compact Artificial Bee algorithm (EcAB)(Banitalebi et al., 2015), compact Bat Algorithm (cBAT) (Dao et al., 2014b), compact Flower Pollination Algorithm (cFPA) (Dao et al., 2016), compact Cat Swarm Optimisation algorithm (cCSO) (Zhao et al., 2017)(Zhao, 2020), compact Firefly Algorithm (cFA) (Tighzert et al., 2018), compact Teaching-Learning-Based Optimisation Method (cTLBO) (Yang et al., 2014b) (Yang et al., 2018), compact Pigeon-Inspired Optimisation (CPIO) (Tian et al., 2020) and the compact Cuckoo Search (cCS) (Song et al., 2020).

- Other algorithms: some compact music-inspired algorithms (Tighzert et al., 2017) (Lachouri et al., 2016);

Some common features clearly appear in most compact algorithms. Before discussing the details of the proposed CScDE algorithm, either from the design or implementation point of view, a unified and detailed view of these features is presented.

2.1. Principles of compact algorithms

Compact algorithms offer a useful alternative of metaheuristics to handle the issue of limited memory in certain applications. They may also shorten the computation time, since they operate on a probabilistic model of solutions instead of a whole population. Till now, a large number of compact algorithms have been proposed. They share the same abstract policy, as illustrated in Figure 1.

The first phase allows initialising the probability vector (PV). It constitutes a shared memory among the future-generated solutions. The recombination between the different solutions is carried out indirectly via this vector. The research phase consists of two operations. In the first operation, which is the generation activity, one or more solutions are sampled in the search space using the PV. The second operation, the selection activity, selects a solution to be retained, this could be the best solution. This solution, maintained at each iteration, biases the PV to converge since it participates in its update. This adjustment would affect the generation of new solutions. Finally, some rules are used in the phase of updating the PV. These phases are repeated until a stop criterion is verified. Algorithm 1 illustrates the high level template of a compact metaheuristic.

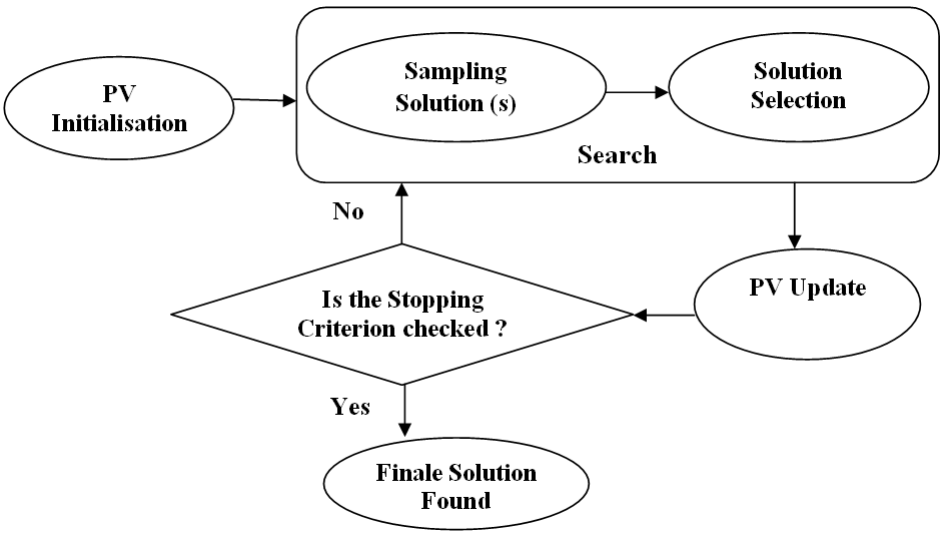


Figure 1: Main phases of compact algorithm.

2.2. A probabilistic model for solution generation

The use of the probabilistic representation allows a reduced use of memory when the algorithm is run. From the literature we consulted, only the binary model of (Harik et al., 1999) and the Gaussian model of (Mininno et al., 2008) were initially developed. Since these models have been the most used ones, special attention is given to them in the next section. Another alternative based on the Gaussian model has been later proposed (Banitalebi et al., 2015). Recently, a fourth model, based on uniform distribution has been introduced (Tighzert et al., 2017). All these models use what is called the probability vector, generally labelled by PV.

To be consistent, we use the following notation throughout the paper: PV for the Probability Vector (or Perturbation Vector), that helps in generating the virtual population which has the size N_P, D and it indicate the problem dimension and the index of iteration, respectively.

Algorithm 1 High level template of compact algorithm.

Input: Probabilistic model PV. Problem dimension D.
Output: *elite*.
generate *elite* by means of PV
while not Termination Criterion **do**
  Generate candidate solution *ind*
  Compare fitness of both solutions
  if *elite* condition replacement is satisfied **then**
    elite ← *ind*
  end if
  Update PV
end while

2.2.1. The binary representation of solutions

The idea of using the probability vector PV instead of a whole population of solutions was originated in the cGA proposed in (Harik et al., 1999). The authors there used a vector PV of a length equal to the dimension of the problem, D. Each element of PV contains the probability that a sample takes the value 1. If the trial is not successful the sample will be set to 0 at the corresponding dimension. All positions of PV are initialised to 0.5 to simulate the usual uniform random initialisation of the basic GA. This model has been used in several compact algorithm proposals based on the binary representation of solutions, including: (Zhou et al., 2002), (Ahn & Ramakrishna, 2003), (Gallagher et al., 2004), (Rimcharoen et al., 2006), (Silva et al., 2007) and (Phiromlap & Rimcharoen, 2013).

2.2.2. The real representation of solutions – The Gaussian model

In most compact algorithms with real encoding presented in the literature, the so-called perturbation vector is actually a $2 \times D$ matrix: $PV = [\mu, \sigma]$, where: μ and σ are vectors containing, for each design variable, the mean and standard deviation values of a Gaussian Probability Distribution Function (PDF) (Mininno et al., 2008). This PDF is adapted such that the domain of the PDF, (-∞, +∞), is truncated and normalised in [-1, +1] and that its surface remains equal to 1 (the surface under the curve of the PDF). The probability density can be described by:

$$PDF(x_i, \mu_i, \sigma_i) = \frac{e^{\left(\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right)}\sqrt{\frac{2}{\pi}}}{\sigma_i\left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} \tag{1}$$

where $p(x_i)$, characterised by μ_i and σ_i, is the value of the PDF corresponding to the design variable x_i and erf is the error function.

The PDF equation is then used to calculate the corresponding distribution function CDF (Cumulative Distribution Function). Since this function is not expressible by common functions, an approximate value of this function is generally used. The CDF is constructed using Chebyshev polynomials (Cody, 1969), more specifically:

$$CDF = \int_0^1 \frac{e\left(\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right)\sqrt{\frac{2}{\pi}}}{\sigma_i\left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} dx \tag{2}$$

It should be noted that the co-domain of the CDF is [0,1].

- *Initialisation:*

  The initialisation of the virtual population is performed in the following way. For each design variable i, $\mu_i = 0$ and $\sigma_i = \lambda$, where: λ is set to a large positive constant (generally, λ is set to 10). This initialisation of σ_i values is done in order to initially have a truncated Normal distribution with a wide shape, to simulate a uniform distribution.

- *The sampling mechanism:*

  A truncated Gaussian PDF with a mean value, μ_i and a standard deviation σ_i, is associated with each standardised design variable indexed by i. Then, the generation of x_i associated to a generic candidate x is produced from the PV(μ_i, σ_i) by following the sampling mechanism described as follows. First, a random number from a uniform distribution $r = rand(0,1)$ is generated. Then, the inverse function of the CDF corresponding to this random value is calculated: $x_i = CDF^{-1}(r)$. As mentioned previously, sampling is done on normalised values in [-1, 1]. Then, to obtain the value of x_i in the original decision space [a, b], denoted $newx_i$, the following operation is performed:

$$newx_i = \frac{(x_i + 1)}{2}(b - a) + a \tag{3}$$

- *Selection and Updating rules*

  When two individuals compete for survival, one that has a better value of the cost function is considered the *winner* while the other is declared to be the *loser*. The *winner* biases the PV by adjusting the mean μ and standard deviation σ by two specific update rules given by:

$$\mu_i^{it+1} \leftarrow \mu_i^{it} + \frac{1}{N_P}(winner_i - loser_i) \tag{4}$$

$$\sigma_i^{it+1} \leftarrow \sqrt{(\sigma_i^{it})^2 + (\mu_i^{it})^2 - (\mu_i^{it+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)} \tag{5}$$

  More details about the construction of these two formulas can be found in (Mininno et al., 2008). By applying this model, only the vectors μ and σ must be stored in memory. This approach has been used in many compact algorithms such as: rcGA (Mininno et al., 2008), cDE (Mininno et al., 2011), cPSO (Neri et al., 2013b), cABC (Dao et al., 2014a), cHS (Tighzert et al., 2017), cTLBO (Yang et al., 2018), cFA (Tighzert et al., 2018).

It is worth to mention here that in compact algorithms, the virtual population size N_p is just a parameter that is used to adapt the value of PV during the search process; there is no population of solutions, with contrast to the population-based algorithms case, where the population size indicates the genuine number of solutions. In other words, if N_p is set to 300, this doesn't mean that there are actually 300 individuals, but rather it represents a step size ($1/N_p$) that controls the adjustment speed of the probabilistic model towards recently sampled good solutions (winners) in such a way that the algorithm operates in an optimal configuration. If the step size is too large, the adjustment is likely to adapt to unfeasible genes of the winners solutions and the algorithm may not behave in a reliable manner. That's because the truncated Gaussian distributions tend to quickly shrink around the newly detected promising solutions. On the contrary, if it is too small, the adaptation will be

too slow and takes very long. That is because new promising solutions have a modest effect on the shrinking of the width of the truncated Gaussian distribution.

It is also important to know that two other distributions have been proposed in the literature such that the memory required for the implementation of a compact algorithm using either one is still the same as the basic model. The first one is an improvement of the Gaussian distribution. It is characterised by a couple of density PDFs that share the same parameters μ and σ which are assigned for each decision variable. As stated in (Banitalebi et al., 2015), this model extends the original one. It was used in EcABC(Banitalebi et al., 2015) and cFPO (Dao et al., 2016). Regarding the second model, it is a *uniform distribution*. It is characterised by adding two new formulas that modify the bounds. Unlike the PDF of the Gaussian distribution, the bounds a and b of this distribution change if its mean and its standard deviation change. More details could be found in (Tighzert et al., 2018).

2.3. Elitism in compact algorithms

Elitism selection is based on the idea of transferring the best solution of the current generation to the next one. When the best current solution (elite) is kept until a better one is found, this is called persistent, permanent or strong elitism. Non-persistent, or non-permanent or weak elitism consists of relaxing the selection pressure by limiting the length of the elite chromosome's inheritance, it is noted η. This means that the elite is updated when another better performing solution does not appear after a given period of time. This strategy diminishes the possibility of the undesirable phenomenon of premature convergence. The parameter η depends on the size of the simulated population (Mininno et al., 2008).

2.4. The stop criterion

The same stop criteria as population-based metaheuristics can be used, i.e., to reach a fixed number of iterations or the maximum number of objective function calls. It is important to mention that in a compact algorithm with a binary coding, as in the cGA and its variants, the algorithm also stops when all the probabilities of the probabilistic vector PV converge to zero or one.

2.5. Basic algorithms

2.5.1. Binary Compact Genetic Algorithm: cGA

The compact genetic algorithm, initially proposed in (Harik et al., 1999), is the simplest version of the "Estimation of Distribution Algorithms" (EDAs) family of algorithms whose main purpose is to simplify the genetic algorithm. It mimics the one-order behaviour of a single GA (sGA) that uses a binary representation and a uniform crossover operator with a reduction of its memory requirements. The progress of the algorithm is carried out as follows. The cGA first starts by initialising each element of the Probabilistic Vector (PV) with the value 0.5. This value indicates the probability of assigning 1 to the i^{th} gene of a chromosome of size D. Then, at each generation during evolution, only two individuals *ind1* and *ind2* are randomly generated according to the probabilities in the current PV. It should be noted that in the first generation, each gene is generated randomly, 0 or 1, with the same probability. The fitness values are calculated for each generated binary chain, and then compared. The one with the best cost function is called the *winner*, and the other is called the *loser*. The competition is held then at the level of each gene of the individuals by favouring the winner. This is done so that if the winner's i^{th} gene takes on a value different from that of the loser, the i^{th} element of the probability vector will be increased or decreased by a constant value. The

latter depends on whether the i^{th} gene of the individual winner is 1 or 0. Finally, the final PV itself represents the final solution found by the cGA. The pseudo code of the cGA is shown in Algorithm 2.

Algorithm 2 Pseudo-code of compact Genetic Algorithm (cGA).

Input: PV, D
Output: PV
Parameters: N_p
for $i \leftarrow 1$ to D **do**
  PV[i] $\leftarrow$0.5
end for
generate two individuals ind_1 and ind_2 by means of PV
while (PV[i] > 0 and PV[i] < 1) **do**
  $[winner, loser] \leftarrow$ compete(ind_1, ind_2)
  for $i = 1 : D$ **do**
    if $winner$[i] $\neq$ $loser$[i] **then**
      if $winner$[i] == 1 **then**
        PV[i] $\leftarrow$ PV[i]+$\frac{1}{N_P}$
      else
        PV[i] $\leftarrow$ PV[i]-$\frac{1}{N_P}$
      end if
    end if
  end for
end while

2.5.2. Real-valued compact differential evolution: cDE

Compact DE (Mininno et al., 2011) start by initialising the $2\times D$ PV structure, as described above. A solution is generated from the PV and plays the role of the elite. For sampling, at each iteration it, some candidate solutions are generated. Their number depends on the mutation scheme selected. For example, if the DE/rand/1 mutation is chosen, three individuals x_r, x_s and x_t are generated from the PV. Mutation is then carried out to generate the mutant x'_{off} according to the following equation:

$$x'_{off} = x_t + F(x_r - x_s). \tag{6}$$

where F is a scaling factor that not only controls the length of the exploration vector $(x_r - x_s)$ but also determines how far away from the current point the descendant should be generated. Besides, a crossover between the elite and the mutant x'_{off} is performed to generate the trial offspring x_{off}. Each gene of the individual x'_{off} is exchanged with the corresponding gene of the elite according to a uniform probability as follows:

$$x_{off}[i] = \begin{cases} x'_{off}[i] & \text{if } rand(0,1) \leq Cr \\ elite[i] & \text{otherwise} \end{cases} \tag{7}$$

where: $rand(0,1)$ is a random number between 0 and 1; i is the index of the gene being examined; Cr is a constant value that represents the crossover rate. This strategy is well known as the binomial crossover. It is the most used variant and claimed to be the most promising one.

Finally, the value of the child's (called trial in the standard DE) cost is calculated and compared with that of the elite. This comparison defines the winner and the loser. The formulas that update the Probabilistic Vector are the same as described above. They are, then, applied for subsequent generations. For the sake of clarity, Algorithm 3 summarises a cDE that uses the $rand/1$ mutation strategy, binomial crossover, and persistent elitism. It is labelled: $cDE/rand/1/bin$.

Algorithm 3 Pseudo-code of compact Differential Evolution (cDE) rand/1/bin.

Input: PV, D
Output: $elite$
Parameters: N_P, F, Cr
$it \leftarrow 0$
for $i = 1 : D$ **do**
  $\mu[i] \leftarrow 0$
  $\sigma[i] \leftarrow \lambda = 10$
end for
generate elite by means of PV
while budget condition **do**
  generate 3 individuals x_r, x_s and x_t by means of PV
  compute $x'_{off} \leftarrow x_t + F(x_r - x_s)$
  $x_{off} \leftarrow x'_{off}$
  for $i \leftarrow 1 : D$ **do**
    if $rand(0,1) > Cr$ **then**
      $x_{off}[i] \leftarrow elite[i]$
    end if
  end for
  $[winner, loser] \leftarrow compete(x_{off}, elite)$
  if x_{off} == winner **then**
    $elite \leftarrow x_{off}$
  end if
  for $i \leftarrow 1 : D$ **do**
    $\mu_i^{it+1} \leftarrow \mu_i^{it} + \frac{1}{N_P}(winner_i - loser_i)$
    $\sigma_i^{it+1} \leftarrow \sqrt{(\sigma_i^{it})^2 + (\mu_i^{it})^2 - (\mu_i^{it+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)}$
  end for
  $it \leftarrow it + 1$
end while

2.6. Application areas of compact algorithms

The literature of optimisation contains many applications based on compact algorithms. Table 1 provides a non-exhaustive list of works which we judged relevant. Those samples may provide useful insights into the types of problems that can be solved with compact algorithms. The use of compact metaheuristics in several applications demonstrates their efficiency and effectiveness to solve certain complex problems. It is to highlight that most of these works concentrated on the application of the binary cGA or a new variant of it to solve a specific problem. We will illustrate to the usefulness, or even necessity, of using compact algorithms in real-world application in the motivation part of Section 3.

Table 1: Compact optimisation in various fields of application.

Application Areas	References	The used algorithm
Image recognition	(Silva et al., 2008)	emCGA
Gray image segmentation	(Zhao, 2020)	cCSO
Robotics	(Iacca et al., 2012a)	cDE-light
	(Neri et al., 2011)	DEcDE
	(Lachouri et al., 2016)	cHAS
	(Tighzert & Mendil, 2016)	cFO
	(Tighzert et al., 2018)	a family of real-valued cFA algorithms
Evolvable hardware	(Aporntewan & Chongstitvatana, 2001)	cGA
	(Gallagher & Vigraham, 2002), (Gallagher et al., 2004)	new cGA alternatives by introducing some modifications namely elitism, mutation and resampling
	(Jewajinda & Chongstitvatana, 2008)	Compact cellular GA
Online control design of a boiler	(Neri et al., 2013b)	cPSO
Precision motion control	(Mininno et al., 2011)	cDE
Road traffic management	(Olarthichachart et al., 2010)	cGA
Optimisation of the pipeline network	(Afshar, 2009)	cGA
Container loading	(Gupta & Tiwari, 2012)	Elitism based cGA
Optimisation of parameters	(Al-Dabbagh et al., 2012)	cGA
	(Al-Dabbagh et al., 2013), (Al-Dabbagh et al., 2014)	I-cGA-sDA (Steepest Descent)
Multivariate calibration	(de Paula et al., 2016)	binary cFA
	(Soares et al., 2014)	mCGA
Wireless sensor networks	(Dao et al., 2014a),(Dao et al., 2015)	cABC
Production planning	(Toledo et al., 2013)	HcGA
Selection of characteristics	(Cantú-Paz, 2002)	cGA, ecGA, Bayesian optimisation
Communication	(Al-Dabbagh, 2009)	cGA
	(Azouaoui et al., 2012)	cGA, OCGAD
	(Shakeel, 2010)	cGA
	(Xing & Qu, 2012)	pe-cGA with three extensions
Bioinformatics	(Badr et al., 2008)	modified pe-cGA
	(Huang et al., 2005)	MCGA
Heterogeneous systems/grids	(Sahu & Satav, 2012)	cGA
	(Kumar Singh & Sahli, 2014)	
Web semantics	(Xingsi et al., 2017)	cGA
Embedded systems	(Timmerman, 2012)	cGA
Artificial neural networks	(Yang et al., 2018)	cTLBO
Deep learning	(Paul et al., 2019)	cGA
Hydroelectric power generation	(Tian et al., 2020)	CPIO
Optimal 3D path planning of underwater unmanned submersibles	(Song et al., 2020)	cCS, pcCS

3. The Proposed Compact Compound Sinusoidal Differential Evolution Algorithm

As mentioned in the introduction, the Opposition-based Compound SinDE "OCSinDE" algorithm (Draa et al., 2018) is actually an improved version of the the original SinDE metaheuristic algorithm proposed in (Draa et al., 2015). The former is mainly based on the use of: compound sinusoidal formulas, an extra-operator (OBL) to boost the algorithm's exploration ability and a restart strategy to face the undesirable stagnation phenomenon. The key idea of the sinusoidal adaptation of the DE main parameters is that at each generation, the values of F and CR are defined as follows:

$$F_{it} = \frac{1}{nWaves} \sum_{w=1}^{nWaves} (0.5 * sin(2 * \pi * freq * it/w) + 1) \tag{8}$$

$$CR_{it} = 0.9 + 0.1 * CRinit_{it} \tag{9}$$

where:

$$CRinit_{it} = \frac{1}{nWaves} \sum_{w=1}^{nWaves} (0.5 * sin(2 * \pi * freq * it/w) + 1) \tag{10}$$

where $nWaves$ is the number of sinusoidal waves to be composed, the variable it denotes the current generation and $freq$ is the frequency of the base wave. It is worth mentioning that $nWaves$ is a new parameter added to OCSinDE compared to basic SinDE.

Motivated by the good behaviour of the OCSinDE, as a result of parameter adaptation, and its high competitiveness against many state-of-the-art methods, such as JADE and SHADE, we propose its compact version here. The choice of borrowing only the sinusoidal adaptation of parameters from the OCSinDE and not the whole mechanism (like the OBL and restart operators) could be justified by the fact that some metaheuristic improvements may perturb others if all are applied once, like well detailed in the paper 'Some metaheuristics should be simplified' (Piotrowski & Napiorkowski, 2018).

Devising a compact variant of such an algorithm aims at opening the door to effectively using it in real-world applications, where the constraint of memory-limitation frequently arises. Interesting examples of such applications are those where thousands, or even millions, of variables need to be optimised, they are very large-scale problems. The optimisation algorithms in those applications are run on memory-limited systems.

Let us detail two situations to better illustrate the need for a compact algorithm. One can try to optimise the weights of a Deep Neural Network (DNN) or a Hidden Markov Model (HMM) in double precision on a micro-controller with *8GB RAM* using compact algorithms instead of gradient-based or population-based approaches. Using the latter class of algorithms such as the CMA-ES would be unfeasible due to quadratic space and time complexity. Therefore, an appropriate attention must be paid to the DNN associated learning method to be effectively implemented on a a micro-controller. In addition, it was already demonstrated in two recent works (Deb & Myburgh, 2016, 2017) that using an evolutionary algorithm (Population-Based Integer Linear Programming: PBILP) capable of handling a scheduling optimisation problem with an astonishing number of variables, one billion, is very memory consuming. It was in the order of **120 GB of RAM**. This amount of memory resources is pretty massive and not permissible in small and cheap built-in devices, compared to compact algorithms (Ferigo & Iacca, 2020).

The compact version of OCSinDE, proposed here, is slightly different from the original one. In particular, we modified Equation (9) as follows:

$$CR_{it} = \{0.7, 0.6\} + 0.1 * CRinit_{it} \tag{11}$$

which forces the value of CR in the interval $[0.6, 0.8]$ instead of $[0.9, 1]$. This range was configured empirically while F parameter remained the same. Figure 2 illustrates both formulas. The key benefits of these formulas, on the one hand, are to take advantage of the periodicity of sine-based functions by preserving the potential to return to certain parameter values even faster than the simple formula given by the original SinDE. On the other hand, they are not built on the maximum number of generations value, which in real-world scenarios is not always known.

Although binomial crossover appears to be more frequently used in many variants of DE or its compact versions, some papers also reported successful use of exponential crossover, such as DEcDE (Neri et al., 2011) and cDE-light (Iacca et al., 2012a). Motivated by these observations, we believe that the use of the exponential crossover needs to be carefully reconsidered, thus, we employed it in our variant. The exponential crossover is applied between the mutant x_{off} and the $elite$ as described in (Neri et al., 2011). A design variable of the provisional offspring x_{off} is randomly selected and copied into the i^{th} design variable of the elite solution (its copy) so that we can make sure the elite and offspring have different genotypes. Subsequently, a set of random numbers between 0 and 1 are generated. As long as $rand(0,1) \leq Cr$, the design variables from the provisional offspring (mutant) are copied into the corresponding positions of the elite. When the condition is not satisfied ($rand(0,1) > Cr$), the copy process is interrupted. Thus, all the remaining design variables of the offspring are copied from the parent. For the sake of clarity, the pseudo-code of the exponential crossover is shown in Algorithm 4.

Algorithm 4 Pseudo-code of the exponential crossover. (Neri et al., 2011)

```
x_off ← elite.
i ← round(D· rand(0, 1))
x_off[i] ← x'_off[i]
while rand ≤ CR do
   x_off[i] ← x'_off[i]
   i ← i + 1
   if i == D then
      i ← 1
   end if
end while
```

The global structure of the proposed CScDE is then given in Algorithm 5. The impact of the different parts of the contribution on the overall performances of our proposal and how they separately influence performances will be investigated in section 4.6.

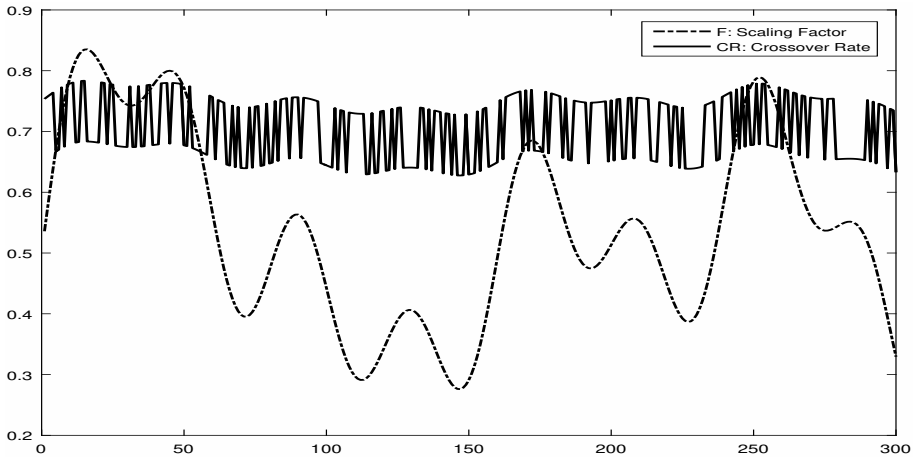


Figure 2: The proposed CScDE parameter setting formulas put side by side.

Algorithm 5 The proposed compact Compound Sinusoidal Differential Evolution Algorithm (CScDE).

Input: PV, D
Output: $elite$
Parameters: N_p, F, Cr
$it \leftarrow 0$
for $i \leftarrow 1$ to D **do**
  Initialise μ
  Initialise σ
end for
generate $elite$ by means of PV
while budget condition **do**
  generate 3 individuals x_r, x_s and x_t by means of PV
  Compute F and Cr using Equations (8), (10), (11)
  compute $x'_{off} \leftarrow x_t + F(x_r - x_s)$
    $x_{off} \leftarrow x'_{off}$
  Apply exponential crossover shown in pseudo-code (4) to generate x_{off}
  $[winner, loser] \leftarrow$ compete$(x_{off}, elite)$
  if $x_{off} = winner$ **then**
    $elite \leftarrow x_{off}$
  end if
  for $i \leftarrow 1$ to D **do**
    $\mu_i^{it+1} \leftarrow \mu_i^{it} + \frac{1}{N_P}(winner_i - loser_i)$
    $\sigma_i^{it+1} \leftarrow \sqrt{(\sigma_i^{it})^2 + (\mu_i^{it})^2 - (\mu_i^{it+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)}$
  end for
  $it \leftarrow it$+1
end while

4. Experimental analysis and discussion

Three sets of problems with various dimensions are considered in our experimentation to investigate the efficiency of the proposed algorithm. The first one is the Black-Box Optimisation Benchmarking (BBOB) (Hansen et al., 2012) test-bed, where the COCO platform is used to facilitate results post-processing, comparisons and illustrations. It is composed of 24 test problems. The second set concerns the CEC-2014 benchmark suite (Liang et al., 2013), composed of 30 functions. We finally use the first five real-world optimisation problems taken from the CEC-2011 benchmark (Das & Suganthan, 2010). Thus, a total of 59 benchmarks is considered.

4.1. Parameter setting

The state-of-the-art compact algorithms considered in this study, to evaluate the merit of the devised algorithm, are listed below with their respective parameter settings.

- real compact Genetic Algorithm (rcGA) (Mininno et al., 2008) with persistent elitism.
- compact Particle Swarm optimisation (cPSO) (Neri et al., 2013b).
- compact Differential Evolution (cDE) (Mininno et al., 2011) with persistent elitism, $rand/1$ mutation and binomial crossover.
- compact Differential Evolution (cDE_Exp) (Mininno et al., 2011) with persistent elitism, $rand/1$ mutation and exponential crossover (Neri et al., 2011).
- Disturbed Exploitation compact Differential Evolution (DEcDE) (Neri et al., 2011) with trigonometric mutation and exponential crossover.
- compact Teaching-Learning Based optimisation (cTLBO) (Yang et al., 2014b)(Yang et al., 2018) characterised by the absence of algorithm parameters that need to be tuned in the optimisation process.
- compact Firefly Algorithm (cFA) (Tighzert et al., 2018) with persistent elitism.

For a fair comparison, all compact algorithms use the same experimental parameters as reported in their original articles. The common parameters are tuned as follows. The virtual population size N_p is set to 300. This value is the one used in all compact algorithms except for cFA in which we used $N_p = 10 * D$, as recommended in the original paper (Tighzert et al., 2018). It is important to stress that, the choice of N_p is important for compact algorithms and it is different from the real size of the population in metaheuristics optimisers. Of note, we didn't carry out an analysis of this parameter but instead, we used the same value adopted for most compact algorithms. We also sat σ to 10 for all algorithms under study as recommended in (Mininno et al., 2008).

Concerning the parameters of the proposed CScDE algorithm, the base frequency of the sinusoidal formula used is set to $freq = 1/D$ and the number of waves being composed is set to $Log(D)$. All parameter configurations are reported in Table2.

All experiments have been conducted with a machine having an I7 2.4 GHz CPU and 16 GB of Memory. As software configuration, we used MATLAB-2015 on a Windows 8 operating system.

Table 2: Parameter setting for compact algorithms used in our experimentation.

CScDE	cDE	cDE-Exp	cPSO	DEcDE	cFA
freq = $1/D$	F = 0.5	F = 0.5	W = -0.2	M_t= 0.003	α = 0.25
nWaves= $Log2(D)$	CR = 0.3	α_m = 0.25	C_1 = -0.07	M_p = 0.001	α_{Dump}=0.995
		$CR = \frac{1}{\alpha_m \sqrt[D]{2}}$	C_2 = 3.74	F = 0.5	β_0= 0.2
				T = 0.1	γ = 1
				α_m = 0.25	
				$CR = \frac{1}{\alpha_m \sqrt[D]{2}}$	

4.2. Comparison of CScDE to state-of-the-art compact algorithms on the BOBO functions

The BBOB test-bed consists of 24 benchmark problems, belonging to five different classes; characterised by different real-world problem features, namely: separable ($F1$-$F5$), moderate ($F6$-$F9$), ill-conditioned ($F10$-$F14$), multimodal ($F15$-$F19$), and weakly-structured multimodal functions ($F20$-$F24$). The actual search space of all functions is $[-5,5]^D$. Details of benchmark functions are described in Appendix D.1.

To have a statistically significant comparison, the algorithms are run over 15 instances (according to the framework) of each benchmark problem (function) in each dimension in order to mitigate the randomness encountered during the test, on a fixed budget of function evaluations equal to $5000 \times D$, being D the problem dimension. It should be noted that all algorithms are tested on the two high dimensions provided by COCO framework, i.e., $D = [20,40]$ (According to the developers of the benchmarking platform, dimension 40 is optional). The termination conditions are achieved by acquiring the maximum number of function evaluation (budget) or getting a solution with a fitness value greater than 10^{-8}. The main measure of performance used here is the Expected Running Time (ERT) described in (Hansen et al., 2016). This metric, used in Figures 3-4, depends on a given target function value, $f_t = f_{opt} + \Delta f$, and it is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{opt}, summed over all trials and divided by the number of trials that actually reached f_t (Hansen et al., 2012).

We start the comparison by confronting our proposal to the state-of-the-art compact algorithms. Figures 3 and 4 graphically illustrates the obtained results, where the best "2009" data plot corresponds to the best observed ERT during BBOB 2009 for each single instance; i.e., it is not an actual competing algorithm, it is the best ever recorded solution over all times for all algorithms. As well seen from the ECDF graphs for dimensions 20 and 40, respectively, CScDE globally outperforms other compact algorithms when the functions considered in this study (F1-F24) are taken as a whole: well seen from the first subplot of Figure 3 (functions 1-24). The performance gap increases as the dimension increases. The superiority of our algorithm becomes more remarkable for the case of separable functions (F1-F5) either for 20 or 40 dimensions. Another important finding here is that the cDE variant based on the exponential crossover, cDE_Exp, marked clear superiority over the binomial crossover-based cDE on all functions and for both dimensions.

For dimension 20, CScDE, cDE_Exp and DEcDE are significantly better than the other algorithms in dealing with moderate functions (F6-F9), ill-conditioned functions (F10-F14), and Weakly-structured multi-modal functions (F20-F24). One can also notice that all compact algorithms perform poorly in multi-modal functions (F15-F19). As dimension 40 is investigated, we see that most of the algorithms fail, particularly on moderate functions and multi-modal functions. However, CScDE has the best performance on ill-conditioned functions and slightly better than DEcDE on moderate

functions.

We could also notice that our CScDE exhibits similar performance to DEcDE to solve the problems of the Weakly-structured multi-modal category, either for D=20 or D=40. Thus, they seem to be efficient alternatives and very promising with respect to other compact algorithms. Another thing that has to be noticed is the disparity in efficiency between the compact Evolutionary Algorithms (cEA) and the compact Intelligence Swarm (cIS), where the former class proved to be superior to the latter. Finally, we disregard the anytime performance of optimisers during the search process on BBOB functions since CScDE got better results on the whole at the final stages especially for Dimension 40.

4.3. Comparison of CScDE to state-of-the-art compact algorithms on the CEC-2014 benchmark

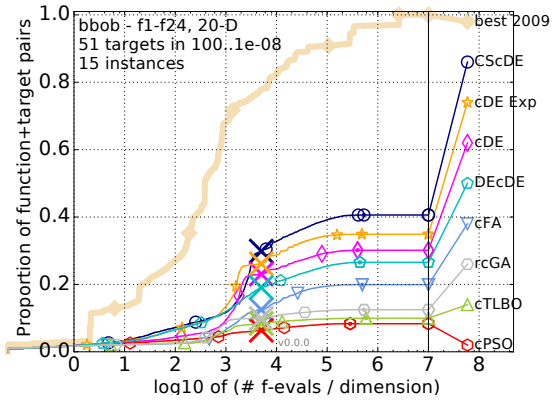
Another suite of 30 well-benchmarked optimisation functions proposed in the CEC-2014 (Liang et al., 2013) special session on real-parameter optimisation is also considered for further assessment of the efficiency of CScDE. The optimisation problems in this benchmark suite are classified into four categories, including three unimodal functions (F1-F3), thirteen simple multimodal functions (F4-F16), six hybrid functions (F17-F22) and seven composition functions (F23-F30). All functions are characterised by being non-separable except for F8 and F10. The main details of these functions are summarised in Table D.26, given in appendix D.

Considering the stochastic nature of the algorithms under comparison, we run each approach 51 times, on each test function with 10, 30, 50 and 100 decision variables, according to the indication in (Liang et al., 2013). The allowed maximum function evaluations (FEs) is similar to the one used previously, i.e., $5000 \times D$ where D is the problem dimensionality.

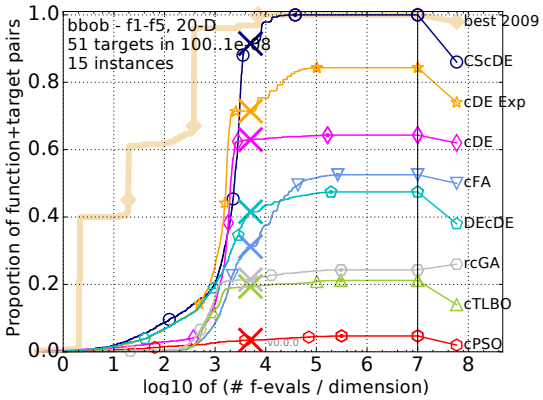
The computational results for benchmark functions with 10, 30, 50 and 100 variables are reported in Tables A.12, A.13, A.14 and A.15 respectively. In all result tables, data are shown in terms of the average fitness error and its standard deviation. The boldface indicates the algorithm that obtained the minimum average error between the best fitness obtained by the algorithms in comparison and the corresponding known optimum on each tested benchmark function. Moreover, the non-parametric Wilcoxon signed-rank test (Wilcoxon, 1992) has been conducted using a significance level of 5% (α = 0.05) between CScDE and each tested algorithm on every benchmark function to investigate the significance of the results. The "+" and "-" symbols tag the cases where CScDE is significantly better and worse than its competitor, respectively, while "=" means that there is no significant difference between their performance. For the sake of clarity, a compact representation of the main experimental results is reported in Table 3 where the last row "+/-/=" summarises the overall comparative results.

Based on the obtained numerical results, CScDE fails to surpass the other cDE variants in dimension 10 on most functions, but it outperforms rcGA, cPSO, cFA and cTLBO. Also, cDE with binomial crossover performs better when tackling low dimension problems compared to cDE variants that employ the exponential crossover i.e. CScDE, cDE_Exp and DEcDE. When the dimension increase to 30, CScDE is superior to rcGA, cDE, cFA, cPSO and cTLBO while DEcDE and cDE_Exp show respectable performance.

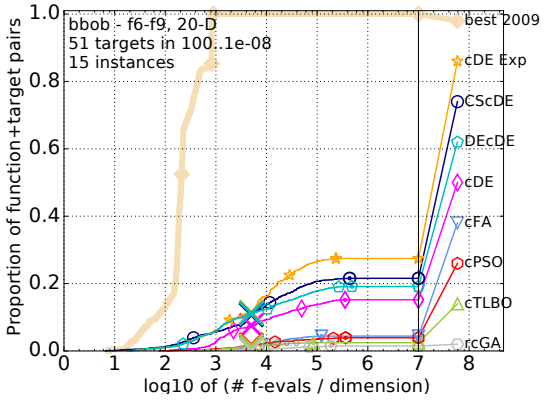
When CEC-2014 problems are scaled up to the dimensionalities 50 and 100, it can be observed that CScDE become more efficient and reveals a marked difference in terms of performances. CScDE is superior to all other cEA and cSI algorithms on most test functions. More specifically, when dimension 100 is taken as example where CScDE shows the best performance, among unimodal functions (F1 – F3), CScDE performed significantly better on F1 and F2, and it is inferior to DEcDE on F3. For multimodal functions (F4 – F16), CScDE finds the best results on F4, F5, F6, F8, F10,



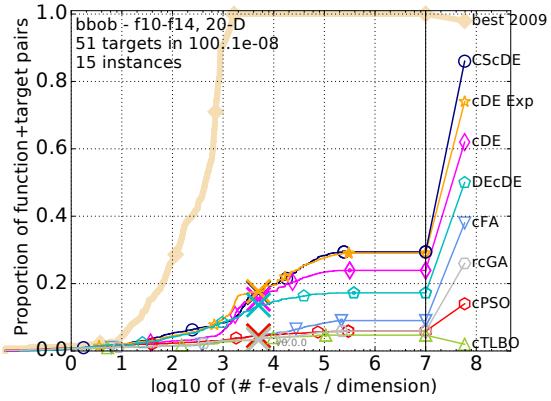
(a) All Functions: F1-F24



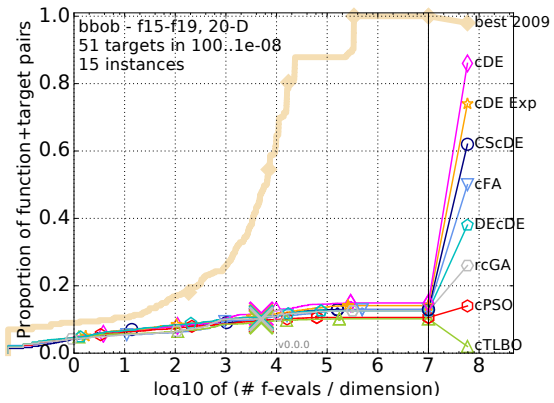
(b) Separable functions: F1-F5



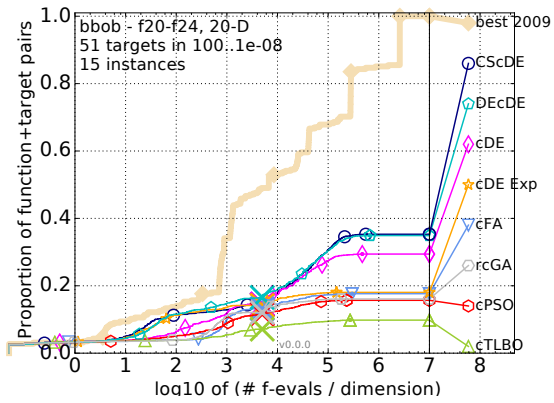
(c) Moderate functions: F6-F9



(d) ill-conditioned functions: F10-F14

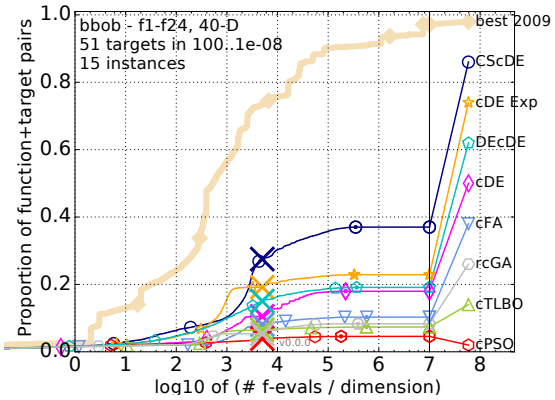


(e) Multi-modal functions: F15-F19

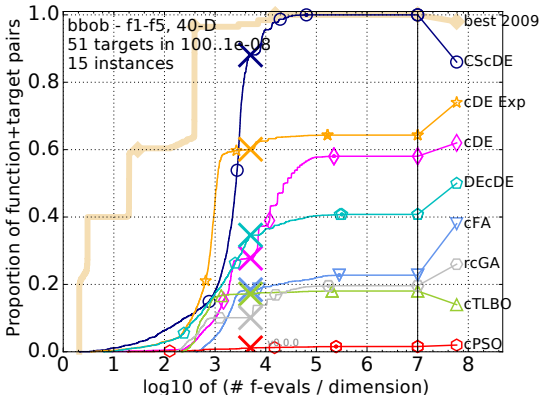


(f) Weakly-structured multi-modal functions: F20-F24

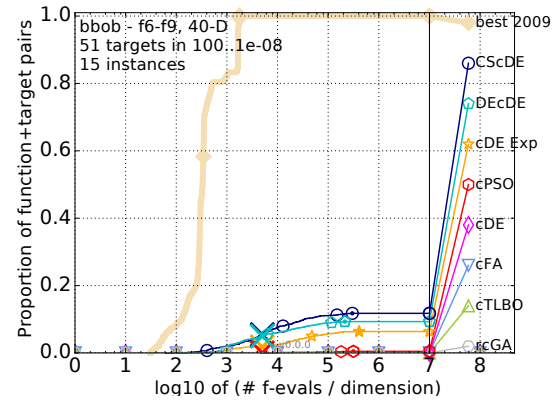
Figure 3: Comparing CScDE to other compact algorithms in 20D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.



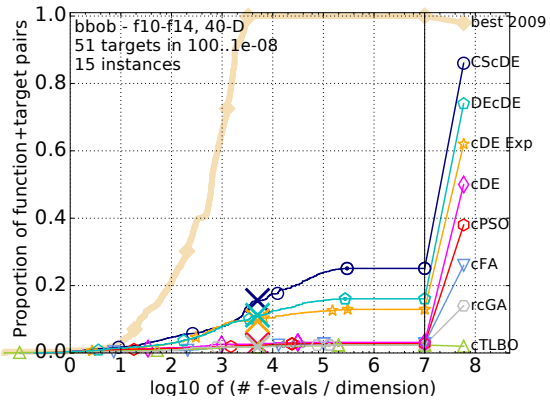
(a) All Functions: F1-F24



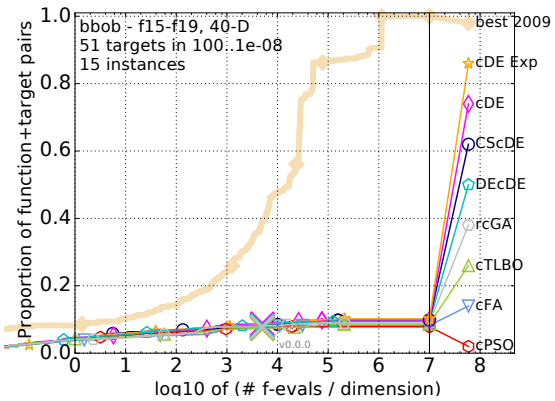
(b) Separable functions: F1-F5



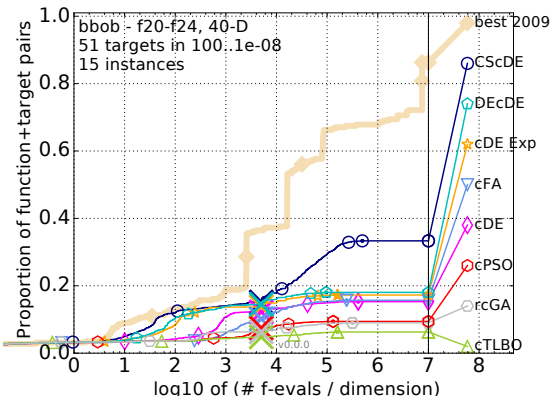
(c) Moderate functions: F6-F9



(d) ill-conditioned functions: F10-F14



(e) Multi-modal functions: F15-F19



(f) Weakly-structured multi-modal functions: F20-F24

Figure 4: Comparing CScDE to other compact algorithms in 40D: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension ($FEvals/D$) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups. As reference algorithm, "best 2009" algorithm from BBOB 2009 is shown as light thick line with diamond markers.

Table 3: The number of statistically significant wins (+), losses (-) and draws (=) of CScDE against rcGA, cDE, cDE_Exp, DEcDE, cFA, cPSO and cTLBO on the CEC-2014 benchmark (Liang et al., 2013) in 10, 30, 50 and 100 dimensions.

	10D	**30D**	**50D**	**100D**	**TOT (+/-/=)**
rcGA	19/2/9	25/1/4	29/0/1	30/0/0	103/3/14
cDE	1/8/21	21/3/6	23/2/5	29/0/1	74/13/33
cDE_Exp	0/3/27	12/2/16	21/0/9	27/0/3	60/5/55
DEcDE	6/4/20	12/5/13	20/2/8	24/2/4	62/13/45
cFA	21/2/7	25/1/4	27/1/2	29/0/1	102/4/14
cPSO	23/4/3	27/1/2	28/0/2	30/0/0	108/5/7
cTLBO	24/0/6	29/0/1	28/0/2	30/0/0	101/0/9

F9, F12, F13, F15 and F16. On F11 and F14, DEcDE and cDE_Exp offer almost the same results as CScDE respectively. On hybrid functions (F17 – F22), while CScDE outperformed by DEcDE on F20 function, it surpasses the other algorithms including DEcDE on F17, F18 (with large advance), F19 and F21. DEcDE performs almost the same way as CScDE on F20 and F22 (as well as cDE_Exp). For composition functions, CScDE is superior to other compared algorithms on F23-F30 except F28 where the algorithm is outperformed by cDE. In addition, one can notice that on F23, F25 and F26, DEcDE achieve nearly the same results as CScDE.

The overall comparison results exhibit that among the 120 cases, CScDE is better/worse than rcGA, cDE, cDE_Exp and DEcDE on 103/3, 74/13, 60/5 and 62/13 cases, respectively. CScDE outperforms cFA, cPSO and cTLBO on 102, 108 and 111 out of 120 cases.

In addition to the results presented above, the ranking among all the algorithms has been performed by means of the Holm-Bonferroni procedure (Holm, 1979) (García et al., 2009) for the eight algorithms to ultimately deduce which one is better. The output results are the one appearing in Table 4 where the rank of CScDE is shown in parentheses. It is also indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected", i.e., CScDE statistically outperforms the algorithm under consideration, or "Not rejected" if the distribution of values can be considered the same (there is no outperformance).

Table 4: Holm-Bonferroni procedure for the CEC-2014 benchmark (Liang et al., 2013) on all algorithms under consideration (reference algorithm CScDE, Rank = 6.76E+00).

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	DEcDE	6.49E+00	-8.43E-01	2.00E-01	5.00E-02	Not rejected
2	cDE_Exp	5.93E+00	-2.64E+00	4.20E-03	2.50E-02	Rejected
3	cDE	5.56E+00	-3.79E+00	7.39E-05	1.67E-02	Rejected
4	cFA	3.82E+00	-9.30E+00	6.87E-21	1.25E-02	Rejected
5	rcGA	2.93E+00	-1.21E+01	4.04E-34	1.00E-02	Rejected
6	cPSO	2.77E+00	-1.26E+01	7.91E-37	8.33E-03	Rejected
7	cTLBO	1.82E+00	-1.56E+01	2.39E-55	7.14E-03	Rejected

On the basis of the results appearing in Table 4, CScDE ranks the best when all dimensions are considered together. It is rather clear now that CScDE is superior to all other algorithms in the

comparison except DEcDE, where the hypothesis is not rejected. In fact, DEcDE is an algorithm in which a perturbation of The PV that behaves similarly to a restart strategy is incorporated in the algorithmic backbone structure. This is not the case for our CScDE. Thus, CScDE would be more tightly integrated with other algorithms if we intend to use it in a hybrid fashion.

The conclusions that can be reached from this analysis on BBOB and CEC-2014 functions are the followings:

1. An inherent limitation of all compact optimisation algorithms including CScDE on multi-modal BBOB functions. In fact, as they lack an actual population of candidate solutions, they cannot guarantee a proper level of diversity, especially in the long run once the Gaussian model has converged. Therefore, unless one introduces some restart mechanisms (as done in some previous works e.g. (Iacca & Caraffini, 2020)), this kind of algorithms are especially good at exploitation but less at exploration, which is what is needed to handle properly multi-modal functions. In fact, once the Gaussian model has converged, new solutions will be sampled in a very small part of the search space, thus making the search local. On the other hand, we should highlight the fact that introducing the sinusoidal mechanism somehow alleviates this effect, compared to previous compact optimisation algorithms from the literature.

2. In the CEC14, CScDE performs poorly on $D = 10$ and remarkably well on $D = 100$. One possible explanation for this -somehow surprising- finding has been given in a recent study by (Caraffini et al., 2017) who collected evidence on the fact that the correlation between pairs of variables appears, from the perspective of a stochastic search algorithm, to consistently decrease when the problem dimensionality increases. In other words, non-separable problems in high dimensionalities can be tackled as if they are separable. The same authors conjecture that this effect is due to fact that in high dimensionalities only a very restricted portion of the decision space can be explored: because of this "localness" of the search, the best strategy to make use of the available budget is to exploit any improvement along each variable, which is in accordance with the most popular and successful methods for large-scale optimisation.
   This might explain then why the proposed Compact Compound Sinusoidal Differential Evolution –which inherently handles each variable separately, as most of the compact optimisation algorithms– works especially well in higher dimensionalities. In fact, the CEC14 problems are mostly non-separable, yet as mentioned in larger dimensionalities can be, in principle, efficiently handled *as if* they are separable. This is also in line with other works on compact algorithms, such as the compact Particle Swarm Optimisation (Neri et al., 2013b). On the contrary, in lower dimensionalities it is always preferable to handle non-separable problems with proper algorithmic moves that handle multiple variables at a time (this is the case of some of the compared algorithms), which again might explain why the proposed CScDE works poorly in 10D.

4.4. Comparison of CScDE to state-of-the-art compact algorithms on CEC-2011 real-world problems

The objective of this section is to test the performances of the proposed algorithm when engaging CEC-2011 real-world problems (Das & Suganthan, 2010) (a more precise description of each one can be found in Table D.27). In accordance with the CEC-2011 rules, a total run number of 25 and a number of evaluation function calls equal to 150000 were used. The compact algorithms, including ours were, tested on five problems exhibiting various dimensions and characteristics. Tables 5 and 6) provide the results in terms of the average fitness, the Wilcoxon test and the outcome of the Holm-Bonferroni procedure, respectively.

In terms of the average fitness, CScDE tends to be a little more successful on the chosen problems than the other algorithms. Table 5 shows that the CScDE algorithm produced the best mean value for three of the five problems (T2, T3 and T5) while it is beaten by DEcDE on one problem T1. For the T4 problem, all eight algorithms produced the same results. In fact, CScDE was the only algorithm able to obtain the lowest fitness on problem T3. This can be partially explained by the stochastic nature of the formula used in CScDE which allows it to achieve better results.

According to the pairwise Wilcoxon Rank-Sign test, CScDE outperforms the other compact algorithms in most cases. On another side, with the Holm-Bonferroni procedure, CScDE ranks second after DEcDE and comes before all other algorithms. Also, the null-hypothesis of statistical equivalence is not rejected on the comparison between CScDE and the other algorithms.

4.5. Comparison of CScDE to population-based metaheuristics on CEC functions

This section is intended to compare the proposed approach (CScDE) to some state-of-the-art population-based metaheuristics with the same budget used previously. The optimisers have been selected according to the availability of the raw data. We have made a distinction between two groups of algorithms: those proposed in CEC-2014 and more recent ones proposed in CEC-2017. Besides, we need to note that we opted for CEC instead of BBOB functions since the experimental results are available for all dimensions, which is not the case for BBOB problems where not all algorithms have their data available for dimension 40.

Of great importance is to highlight that compared to population-based metaheuristics it is unlikely that low complexity compact algorithms, that use only 2-3 solutions, offer superior performances, but in some specific contexts, the latter class of metaheuristics may give very satisfactory results. This is shown in the following comparisons.

4.5.1. Comparison of CScDE to CEC-2014 population-based metaheuristics

We choose from the CEC-2014 competition the following three algorithms:

- Differential Evolution with Rotation-Invariant Mutation and Competing Strategies Adaptation: b3e3pbest (Bujok et al., 2014);
- Linear Population Size Reduction SHADE: L-SHADE (Tanabe & Fukunaga, 2014);
- Differential Evolution with Replacement Strategy: RSDE (Xu et al., 2014).

Statistically speaking and based on the mean metric (we focused on the average values in order to simplify the interpretation of the outcome), the results show that the CScDE method did not perform very well when all problems and all dimensions are considered, compared to the population-based metaheuristics, see Tables B.16, B.17, B.18 and B.19. It does, however, achieve better solutions than others in many cases. We highlight some examples in which CScDE fits well and outperforms other algorithms, not necessary all of them. For the experiment with 10 dimension, CScDE and the other rivals were beaten by the best performing algorithm but CScDE did relatively well compared to both b3e3pbest and RSD on several functions such as F5, F11, F19 and F29. It performed better that the three other algorithms on F8, F10, F22 and F23. For the experiment with 30 dimensions, CScDE only achieved better results than L-SADE, b3e3pbest and RSDE on F5, F8, F10, F11, F12 and F16.

When 50 and 100 dimensions are considered, CScDE defeated even L-SHADE on many functions. For instance, If we are to sought for better results on F5, F8, F9, F10, F11, F12, F14, F15, F16, F22,

Table 5: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against rcGA, cDE, cDE_Exp, DEcDE, cPSO, cTLBO and cFA on CEC-2011 (Das & Suganthan, 2010)

Function	**CScDE**		**cDE_Exp**			**DEcDE**			**cDE**			**rcGA**			**cPSO**			**cTLBO**			**cFA**		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.80E+01	6.24E+00	1.59E+01	5.22E+00	-	**9.02E+00**	**6.54E+00**	-	1.45E+01	6.94E+00	-	1.95E+01	5.99E+00	=	1.17E+01	4.83E+00	-	1.91E+01	3.93E+00	=	2.19E+01	3.87E+00	+
2	**-2.25E+01**	**3.86E+00**	-2.02E+01	4.21E+00	=	-2.13E+01	4.33E+00	=	-1.68E+01	3.26E+00	+	-6.88E+00	1.91E+00	+	-6.90E+00	8.81E-01	+	-7.63E+00	2.46E+00	+	-8.13E+00	1.71E+00	+
3	**0.00E+00**	**0.00E+00**	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+	1.15E-05	1.73E-21	+
4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
5	**-3.46E+01**	**1.07E+00**	-3.39E+01	1.47E+00	+	-3.41E+01	1.59E+00	+	-3.13E+01	3.33E+00	+	-2.27E+01	4.12E+00	+	-2.10E+01	1.51E+00	+	-2.13E+01	2.08E+00	+	-2.31E+01	3.77E+00	+
+/-/=			2/1/2			2/1/2			3/1/1			3/0/2			3/1/1			3/0/2			4/0/1		

Table 6: Holm-Bonferroni procedure for five CEC-2011 benchmark (Das & Suganthan, 2010) (reference algorithm DEcDE, Rank = 7.40E+00).

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	CScDE	7.20E+00	-1.29E-01	4.49E-01	5.00E-02	Not rejected
2	cDE_Exp	6.40E+00	-6.45E-01	2.59E-01	2.50E-02	Not rejected
3	cDE	6.20E+00	-7.75E-01	2.19E-01	1.67E-02	Not rejected
4	cPSO	5.00E+00	-1.55E+00	6.07E-02	1.25E-02	Not rejected
5	cFA	4.80E+00	-1.68E+00	4.66E-02	1.00E-02	Not rejected
6	cTLBO	4.60E+00	-1.81E+00	3.54E-02	8.33E-03	Not rejected
7	rcGA	4.20E+00	-2.07E+00	1.94E-02	7.14E-03	Not rejected

F24 and F26, CScDE is the best option compared to L-SHADE and the other algorithms with regard to the case of dimension 100.

In Table 7, we can observe that the reference algorithm is L-SHADE since it is the one with the best rank. The last algorithm is b3e3pbest because it obtained the worst rank. CScDE ranked before the last algorithm and after RSD.

Table 7: Holm-Bonferroni procedure for CEC-2014 benchmark (Das & Suganthan, 2010) (reference algorithm L-SHADE, Rank = 3.29E+00).

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RSDE	2.85E+00	-2.65E+00	4.02E-03	5.00E-02	Rejected
2	CScDE	2.32E+00	-5.85E+00	2.46E-09	2.50E-02	Rejected
3	b3e3pbest	1.61E+00	-1.01E+01	2.76E-24	1.67E-02	Rejected

4.5.2. Comparison of CScDE to CEC-2017 population-based metaheuristics

In a similar and a complementary way of the previous section, we choose from the CEC-2017 competition, the following six algorithms:

- Hybrid LSHADE with Semi-Parameter Adaptation and CMA-ES: LSHADE_SPACMA (Mohamed et al., 2017);
- IPOP-CMA-ES with Midpoint: RB_IPOP_CMAES (Biedrzycki, 2017);
- JSO (Brest et al., 2017);
- Effective Butterfly Optimiser using Covariance Matrix Adapted Retreat phase (Kumar et al., 2017);
- Teaching Learning Based Optimisation with Focused LearningTeaching Learning Based Optimisation with Focused Learning: TLBO_FL (Kommadath & Kotecha, 2017)
- Proactive Particles in Swarm Optimisation: PPSO (Tangherloni et al., 2017)

Tables C.20, C.21, C.22, C.23 and C.24 contain respectively, the results in terms of Average error ± standard deviation, the results when applying the Wilcoxon test and the outcome of the Holm-Bonferroni procedure when all dimensions are considered. Since our algorithm shows remarkable

performance on D = 100, we also applied the Holm-Bonferroni procedure taking into account only dimension 100, see Table 8.To enrich this analysis, we also identify the functions where CScDE outperforms advanced population-based algorithms on the basis of the mean metric as indicated in Table 9.

Our main comments are related to the content of Table 8. We can see that CScDE could come in a better position than the PPSO and TLBO_FL algorithms with regard to dimensionality 100. Table 9 draws our attention to cases where CScDE is still able to find better solutions than other high-quality algorithms, including the highest performin ones such as JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES and EBOwithCMAR. This is another indication of the promises the proposal offers: even with a limited memory space offered, a well-devised compact algorithm could be as powerful as well-established metaheuristics (some of the above-mentioned ones are CEC-2017 winners).

Table 8: Holm–Bonferroni procedure (Reference: LSHADE_SPACMA, Rank = 5.87E+00) for the seven algorithms under consideration over the 30 CEC-2017 benchmark functions for D=100.

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	5.77E+00	-2.07E-01	4.18E-01	5.00E-02	Not rejected
2	EBOwithCMAR	4.23E+00	-3.38E+00	3.61E-04	2.50E-02	Rejected
3	JSO	4.10E+00	-3.66E+00	1.27E-04	1.67E-02	Rejected
4	CScDE	3.13E+00	-5.66E+00	7.63E-09	1.25E-02	Rejected
5	PPSO	2.77E+00	-6.42E+00	6.92E-11	1.00E-02	Rejected
6	TLBO_FL	2.13E+00	-7.73E+00	5.43E-15	8.33E-03	Rejected

Table 9: CEC-2017 benchmark functions on which CScDE outperforms EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO and TLBO_FL in 10, 30, 50, and 100 dimensions.

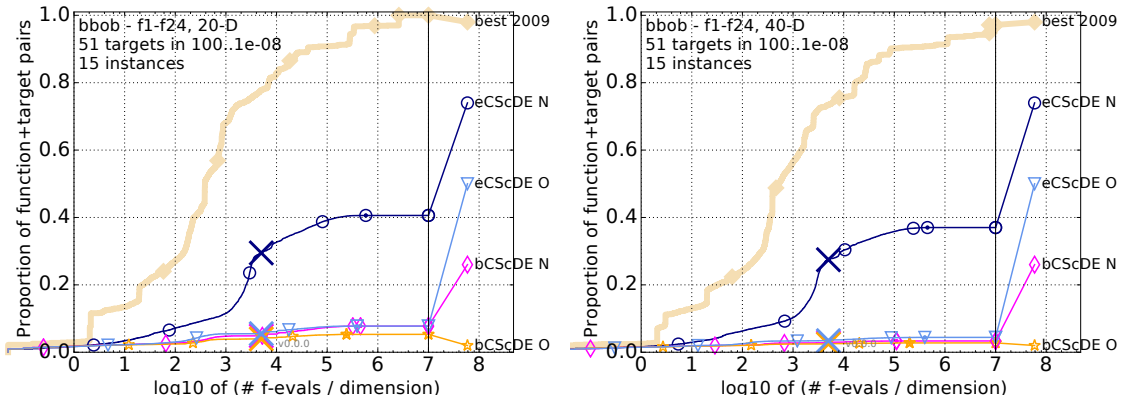
	D10	D30	D50	D100
All algorithms	{∅}	{∅}	{∅}	{∅}
EBOwithCMAR	{∅}	10	10,17,20,23	10,16,17,20,22-24,26
JSO	10,20,24	5,7,8,10,21,23	5,7,8,10,20-23	5,7,8,10,16,17,20-24,26,29
LSHADE_SPACMA	25	10,23	10,21,23	10,17,21,23,24
RB_IPOP_CMA_ES	16,17,20	11	11,12,30	30
PPSO	5,6,11,20,23,27	5-7, 9-11, 21,23,24,27	4-10, 21-24, 26,27,29	1,4-8,12,21-30
TLBO_FL	3,5,7,8,10,17,20,25,28	2,4,6,7,10,11,13,15,19,25,28,30	1,4,6,7,10,11,18,20,22,25,27,28,30	1,2,4,6,7,9,10,12-14,18,20,22-28,30

4.6. Influence of Binomial / Exponential crossover form and New / Old formula on CScDE efficiency

The plots report the comparison of the four variants in terms of ECDF, where they are ranked in an increasing order. If we analyse Figure 5, we can say that combining exponential crossover with the new formula of CR (eCDcDE_N is the same as CScDE) has a positive influence on the performance of our optimiser. That is, the efficiency of both modifications is much higher than the compact constituted with the main components of the original OCSinDE (i.e the binomial crossover and the old formula of CR (shown in Equation (9)). As we have said before, this new formula of CR was fixed based on an empirical process.

4.7. Algorithmic Complexity

Our experimental investigation concludes with a look at the proposed algorithms' time complexity and memory demands.



(a) All Functions: F1-F24 ($D = 20$) (b) All Functions: F1-F24 ($D = 40$)

Figure 5: Influence of Binomial/Exponential crossover form and New/Old formula of CR on CScDE efficiency

4.7.1. Time complexity

First, we use the COCO platform's metrics to measure the algorithm's time complexity (Hansen et al., 2012). The total CPU time is calculated by using the BBOB testbed f8 function (Rosenbrock) over several iterations for at least ten seconds. For each dimensionality, per function evaluation, the CPU time is reported. COCO developers advise to conductor the experiment on time complexity in the same dimensions as benchmarking experiments, but to achieve a more clear estimate of the complexity graph, we have broadened the test to include additional dimensions (10, 20, 30, 40, 50, 100, 300, 500, 700, 800 and 1000). The visual representation of complexity of CScDE, as illustrated in Figure 6, is linear. Second, we calculated the algorithmic time complexity according to the CEC-2017 benchmark specifications. Table 10 summarises the algorithmic complexity of the proposed CScDE algorithm in terms of the $\hat{TC}$ metric on D = 10, 30, 50, and 100 dimensions respectively. More specifically, TC refers to the time to execute the proposed algorithm (with the same parameters described above) with a budget of 200,000 FEs on the $F18$ CEC-2017 benchmark function. $\hat{TC}$ denotes the mean of TC computed over 30 independent runs, in order to provide a more reliable estimation of our algorithm's time complexity. Again, as seen in the table, our algorithm's average CPU time tends to increases linearly. At 100D, for example, the execution time is nearly twice as that of 50D.

Table 10: CScDE time complexity estimation.

Dimension	$\hat{TC}$[s]
10	8.77
30	13.77
50	19.89
100	37.71

Third, we move to estimate the theoretical complexity, in terms of the big Oh notation, of the CScDE algorithm. As seen from the pseudocode of Algorithm 5, the CScDE algorithm is composed of: (1) an elementary instruction of initialisation with a complexity equal to $\mathcal{O}(1)$; (2) a first 'for

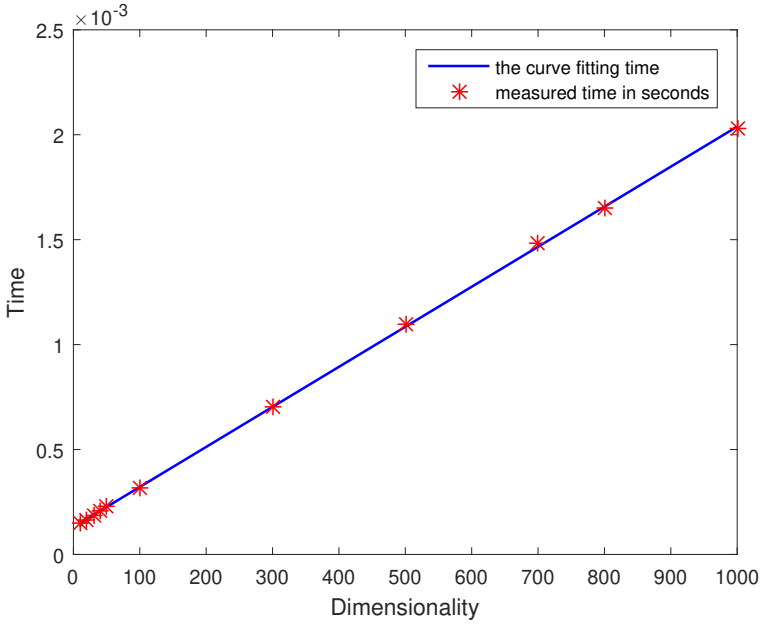


Figure 6: CScDE algorithm complexity estimation using the COCO metrics.

loop', that repeats two elementary instructions D times, D is the problem dimensionality; it has so a complexity equal to $\mathcal{O}(D)$; (3) an instruction that generates the elite by means of PV, it has $\mathcal{O}(D)$ as complexity; and finally (4) a 'while loop', containing itself some elementary instructions. The while condition in the latter bloc has a constant number of repetitions, generally a predefined number of generations. So, its complexity is similar to that of the internal 'for loop', i.e. $\mathcal{O}(D)$. The other instructions inside the while loop have either a $\mathcal{O}(1)$ (basic comparisons or mathematical operations) or $\mathcal{O}(D)$ complexity (like generating the three individuals based on PV). Thus, the whole 'while loop' has a complexity equal to $\mathcal{O}(D)$. Adding all the parts, we find that the CScDE algorithm has a complexity equal to $\mathcal{O}(D)$, with D the problem dimensionality. This is the same complexity we just found using the CEC and BBOB time-measuring metrics.

Hence, the CScDE inherits the linear complexity of the basic DE, SinDE and OCSinDE algorithms from which it was inspired.

4.7.2. Spatial complexity

In this part of the complexity analysis, we will look at memory as the second factor. The same algorithms are compared in terms of the approximate number of D-dimensional vectors used to perform the algorithmic operations in memory. The details in Table 11 indicates that all compact algorithms need significantly less memory space than population-based algorithms.

5. Conclusion and suggestions for future research

This work introduced a new optimisation compact algorithm, called the *compact compound Sinusoidal Differential Evolution Algorithm*. Assisted by a simple configuration of the compound formulas for adjusting F and CR DE parameters proposed for the classical DE, the algorithm is capable

Table 11: Memory complexity of the algorithms analysed in terms of the approximate number of "memory slots" (i.e. D-dimensional vectors)

.

	Algorithm	Number of memory slots
Compact	CScDE	4
	DEcDE	4
	cDE_exp	4
	cDE	4
	cFA	4
	rcGA	4
	cPSO	5
	cTLBO	5
CEC-2014 Population-based	b3e3pbest	$PopSize + ArchiveSize$ = 100 + 100 = 200
	L-SHADE	$PopSize + ArchiveSize + HistoricalMemorySize$ =round(D*rinit)+round(Ninit*rarc)+H =18*D+47*D+6=65*D+6
	RSDE	$PopSize$ + 2 = 52
CEC-2017 Population-based	EBOwithCMAR	$PopSize = PS_1 + PS_2 + PS_3$+HistoricalMemory ($PS_1$+ 64.8*$D$, PS_2 = 4, PS_3= 3*log(D), HistoricalMemory= 6)
	LSHADE_SPACMA	$PopSize$+ MemorySize ($PopSize = 18 * D$, $MemorySize$ = 5)
	JSO	2*$PopSize$+HistoricalMemory ($PopSize$=25*$log(D)\sqrt{D}$, HistoricalMemory= 5)
	RB-IPOP-CMAES	2+$\lfloor$(3/2)*log(D)$\rfloor$+(nbrGenerations-1)*(16+$\lfloor$12*log(D)$\rfloor$)
	PPSO	3*$PopSize$+1 ($PopSize$=200/300/500, according to the dimension D)
	TLBO_FL	$PopSize$+1 ($PopSize$ =100)

of collecting more valuable information to find better-converged solutions, which is an improvement of the compact versions of the traditional DE (i.e cDE/cDE_Exp).

The comparison of the proposed CScDE algorithm to seven state-of-the-art compact algorithms, through the BBOB framework, CEC-2014 functions, and CEC-2011 real-world problems in several dimensions, has proven its superiority on most function groups. CScDE was found to be more successful as the size and complexity of the problem increased. In addition, through comparison with the results of certain studies in the literature, it was demonstrated that the proposed method produced acceptable results on dimensions 50 and 100 than other high-quality algorithms.

In summary, as stated by the No Free Lunch Theorems (NFLT) (Wolpert & Macready, 1997), it is widely agreed that there is no single algorithm being suitable to all optimisation problems in all dimensions. Also, since every approach has its advantages, it is useful to use existing algorithms to create new variants. In this context, we believe that, although the existence of many variants of cDE or other compact algorithms, the NFLT can still apply on the proposed CScDE not to mention that not all metaheuristics can have their compact variant implemented straightforwardly. On the other hand, compared to standard compact algorithms under study, CScDE was found a promising one based on the examined results and the simplicity of the compact structure design. Empirical evidence and the theoretical arguments reflect a clear incentive to continue advancing the compact optimisation field.

For future work, it would be valuable to investigate the effect of different modifications to CScDE to deal with the undesirable phenomena of stagnation. The adjustment mechanism of parameters in CScDE is quite general and hence can be integrated into other compact optimisation algorithms to improve their capability. This would be another interesting research direction.

References

Afshar, M. (2009). Application of a compact genetic algorithm to pipe network optimization problems. *Sceitia Iranica, Transaction A: Civil Engineering, Sharif University of Technology*, *16*, 264–271.

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

Al-Dabbagh, R. (2009). Compact genetic algorithm for cryptanalysis trapdoor 0-1 knapsack cipher. *J. Al-Nahrain Univ*, *12*, 137–145.

Al-Dabbagh, R. D., Baba, M. S., Mekhilef, S., & Kinsheel, A. (2012). The compact genetic algorithm for likelihood estimator of first order moving average model. In *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)* (pp. 474–481). IEEE.

Al-Dabbagh, R. D., Kinsheel, A., Baba, M. S., & Mekhilef, S. (2013). An integration of compact genetic algorithm and local search method for optimizing arma (1, 1) model of likelihood estimator. In *Proceedings of 2nd International Conference on Computer Science and Computational Mathematics* (pp. 60–67).

Al-Dabbagh, R. D., Kinsheel, A., Baba, M. S., & Mekhilef, S. (2014). A combined compact genetic algorithm and local search method for optimizing the arma (1, 1) model of a likelihood estimator. *Sci. Asia*, *40*, 78–86.

Aporntewan, C., & Chongstitvatana, P. (2001). A hardware implementation of the compact genetic algorithm. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)* (pp. 624–629). IEEE volume 1.

Awad, H., N, Ali, Z., M, Qu, Y., B, Liang, J., J, & Suganthan, N., P (November 2016). *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization*. Technical Report Nanyang Technological University, Singapore.

Azouaoui, A., Berkani, A., & Belkasmi, M. (2012). An efficient soft decoder of block codes based on compact genetic algorithm. *arXiv preprint arXiv:1211.3384*, .

Badr, A., Aref, I. M., Hussien, B. M., & Eman, Y. (2008). Solving protein folding problem using elitism-based compact genetic algorithm. *Journal of Computer Science*, *4*, 525–529.

Banitalebi, A., Aziz, M. I. A., Bahar, A., & Aziz, Z. A. (2015). Enhanced compact artificial bee colony. *Information Sciences*, *298*, 491–511.

Biedrzycki, R. (2017). A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1489–1494). IEEE.

Brest, J., Maučec, M. S., & Bošković, B. (2017). Single objective real-parameter optimization: Algorithm jSO. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 1311–1318). IEEE.

Bujok, P., Tvrdík, J., & Polakova, R. (2014). Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2253–2258). IEEE.

Cantú-Paz, E. (2002). Feature subset selection by estimation of distribution algorithms. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (pp. 303–310). Morgan Kaufmann Publishers Inc.

Caraffini, F., Neri, F., & Iacca, G. (2017). Large scale problems in practice: The effect of dimensionality on the interaction among variables. In *European Conference on the Applications of Evolutionary Computation* (pp. 636–652). Springer.

Cody, W. J. (1969). Rational chebyshev approximations for the error function. *Mathematics of Computation*, *23*, 631–637.

Dao, T.-K., Chu, S.-C., Shieh, C.-S., Horng, M.-F. et al. (2014a). Compact artificial bee colony. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 96–105). Springer.

Dao, T.-K., Pan, J.-S., Chu, S.-C., Shieh, C.-S. et al. (2014b). Compact bat algorithm. In *Intelligent Data analysis and its Applications, Volume II* (pp. 57–68). Springer.

Dao, T.-K., Pan, T.-S., Nguyen, T.-T., & Chu, S.-C. (2015). A compact artificial bee colony optimization for topology control scheme in wireless sensor networks. *Journal of Information Hiding and Multimedia Signal Processing*, *6*, 297–310.

Dao, T.-K., Pan, T.-S., Nguyen, T.-T., Chu, S.-C., & Pan, J.-S. (2016). A compact flower pollination algorithm optimization. In *2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN)* (pp. 76–79). IEEE.

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

Deb, K., & Myburgh, C. (2016). Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (pp. 653–660).

Deb, K., & Myburgh, C. (2017). A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. *European Journal of Operational Research*, *261*, 460–474.

Draa, A., Bouzoubia, S., & Boukhalfa, I. (2015). A sinusoidal differential evolution algorithm for numerical optimisation. *Applied Soft Computing*, *27*, 99–126.

Draa, A., Chettah, K., & Talbi, H. (2018). A compound sinusoidal differential evolution algorithm for continuous optimization. *Swarm and Evolutionary Computation*, .

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

Finck, S., Hansen, N., Ros, R., & Auger, A. (2010). *Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions*. Technical Report Citeseer.

Gallagher, J. C., & Vigraham, S. (2002). A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (pp. 163–170). Morgan Kaufmann Publishers Inc.

Gallagher, J. C., Vigraham, S., & Kramer, G. (2004). A family of compact genetic algorithms for intrinsic evolvable hardware. *IEEE Transactions on evolutionary computation*, *8*, 111–126.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, *13*, 959.

Gupta, P., & Tiwari, R. (2012). Solving three dimensional bin packing problem using elitism based genetic algorithm. *Int. J. Adv. Res. Comput. Eng. Technol*, *1*, 471–475.

Hansen, N., Auger, A., Brockhoff, D., Tušar, D., & Tušar, T. (2016). Coco: performance assessment. *arXiv preprint arXiv:1605.03560*, .

Hansen, N., Auger, A., Finck, S., & Ros, R. (2012). Real-parameter black-box optimization benchmarking: Experimental setup. *Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep*, .

Hansen, N., Finck, S., Ros, R., & Auger, A. (2009). *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. Ph.D. thesis INRIA.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE transactions on evolutionary computation*, *3*, 287–297.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, (pp. 65–70).

Huang, Y.-C., Chang, C.-F., Chan, C.-h., Yeh, T.-J., Chang, Y.-C., Chen, C.-C., & Kao, C.-Y. (2005). Integrated minimum-set primers and unique probe design algorithms for differential detection on symptom-related pathogens. *Bioinformatics*,

21, 4330–4337.

Iacca, G., & Caraffini, F. (2020). Re-sampled inheritance compact optimization. *Knowledge-Based Systems*, (p. 106416).

Iacca, G., Caraffini, F., & Neri, F. (2012a). Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead. *Journal of Computer Science and technology*, *27*, 1056–1076.

Iacca, G., Mallipeddi, R., Mininno, E., Neri, F., & Suganthan, P. N. (2011a). Global supervision for compact differential evolution. In *2011 IEEE Symposium on Differential Evolution (SDE)* (pp. 1–8). IEEE.

Iacca, G., Mininno, E., & Neri, F. (2011b). Composed compact differential evolution. *Evolutionary Intelligence*, *4*, 17–29.

Iacca, G., Neri, F., & Mininno, E. (2011c). Opposition-based learning in compact differential evolution. In *European Conference on the Applications of Evolutionary Computation* (pp. 264–273). Springer.

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

Jewajinda, Y. (2016). Covariance matrix compact differential evolution for embedded intelligence. In *2016 IEEE Region 10 Symposium (TENSYMP)* (pp. 349–354). IEEE.

Jewajinda, Y., & Chongstitvatana, P. (2008). Fpga implementation of a cellular compact genetic algorithm. In *2008 NASA/ESA Conference on Adaptive Hardware and Systems* (pp. 385–390). IEEE.

Kommadath, R., & Kotecha, P. (2017). Teaching learning based optimization with focused learning and its performance on CEC 2017 functions. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 2397–2403). IEEE.

Kumar, A., Misra, R. K., & Singh, D. (2017). Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 1835–1842). IEEE.

Kumar Singh, P., & Sahli, N. (2014). Task scheduling in grid computing environment using compact genetic algorithm. *Int. J. Sci. Eng. Technol. Res.(IJSETR)*, *3*.

Lachouri, F., Khelifi, A., Tighzert, L., Aguercif, T., & Mendil, B. (2016). Self-stunding up of humanoid robot using a new intelligent algorithm. In *2016 8th International Conference on Modelling, Identification and Control (ICMIC)* (pp. 903–908). IEEE.

Liang, J. J., Qu, B. Y., & Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, *635*.

Mininno, E., Cupertino, F., & Naso, D. (2008). Real-valued compact genetic algorithms for embedded microcontroller optimization. *IEEE Transactions on Evolutionary Computation*, *12*, 203–219.

Mininno, E., Neri, F., Cupertino, F., & Naso, D. (2011). Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, *15*, 32–54.

Mohamed, A. W., Hadi, A. A., Fattouh, A. M., & Jambi, K. M. (2017). LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In *2017 IEEE Congress on evolutionary computation (CEC)* (pp. 145–152). IEEE.

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

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

Neri, F., & Mininno, E. (2010). Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, *5*, 54–65.

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

Olarthichachart, P., Kaitwanidvilai, S., & Karnprachar, S. (2010). Trip frequency scheduling for traffic transportation management based on compact genetic algorithm. In *Proceedings of International MultiConference of Engineers and Computer Scientists* (pp. 1072–1074). Citeseer.

Paul, S., Singh, L. et al. (2019). Simultaneous structure and parameter learning of convolutional neural network. In *Computational Intelligence: Theories, Applications and Future Directions-Volume II* (pp. 93–104). Springer.

de Paula, L. C., Nogueira, H. V., Soares, A. S., de Lima, T. W., & Coelho, C. J. (2016). A compact firefly algorithm for the variable selection problem in pharmaceutical ingredient determination. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 3832–3838). IEEE.

Phiromlap, S., & Rimcharoen, S. (2013). A frequency-based updating strategy in compact genetic algorithm. In *2013 International Computer Science and Engineering Conference (ICSEC)* (pp. 207–211). IEEE.

Piotrowski, A. P., & Napiorkowski, J. J. (2018). Some metaheuristics should be simplified. *Information Sciences*, *427*, 32–62. URL: https://www.sciencedirect.com/science/article/pii/S0020025517310332. doi:https://doi.org/10.1016/j.ins.2017.10.039.

Prabha, S., & Yadav, R. (2020). Differential evolution with biological-based mutation operator. *Engineering Science and*

Technology, an International Journal, *23*, 253–263.

Prügel-Bennett, A. (2010). Benefits of a population: Five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation*, *14*, 500–517.

Rimcharoen, S., Sutivong, D., & Chongstitvatana, P. (2006). Updating strategy in compact genetic algorithm using moving average approach. In *2006 IEEE Conference on Cybernetics and Intelligent Systems* (pp. 1–6). IEEE.

Sahu, N., & Satav, S. (2012). Task scheduling using compact genetic algorithm for heterogeneous system. *International Journal of Advanced Research in Computer Engineering and Technology*, *1*, 218–221.

dos Santos Coelho, L., Souza, R. C. T., & Mariani, V. C. (2009). Improved differential evolution approach based on cultural algorithm and diversity measure applied to solve economic load dispatch problems. *Mathematics and Computers in Simulation*, *79*, 3136–3147.

Sergio, A., Carvalho, S., & Marco, R. (2014). On the use of compact approaches in evolution strategies. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *3*, 13–23.

Shakeel, I. (2010). Ga-based soft-decision decoding of block codes. In *2010 17th International Conference on Telecommunications* (pp. 13–17). IEEE.

Shang, Y.-W., & Qiu, Y.-H. (2006). A note on the extended rosenbrock function. *Evolutionary Computation*, *14*, 119–126.

Silva, R. R., Lopes, H. S., & Lima, C. R. E. (2007). A new mutation operator for the elitism-based compact genetic algorithm. In *International Conference on Adaptive and Natural Computing Algorithms* (pp. 159–166). Springer.

Silva, R. R., Lopes, H. S., & Lima, C. R. E. (2008). A compact genetic algorithm with elitism and mutation applied to image recognition. In *International Conference on Intelligent Computing* (pp. 1109–1116). Springer.

Soares, A., De Lima, T., Soares, F., Coelho, C., Federson, F., Delbem, A., & Van Baalen, J. (2014). Mutation-based compact genetic algorithm for spectroscopy variable selection in determining protein concentration in wheat grain. *Electronics Letters*, *50*, 932–934.

Song, P.-C., Pan, J.-S., & Chu, S.-C. (2020). A parallel compact cuckoo search algorithm for three-dimensional path planning. *Applied Soft Computing*, (p. 106443).

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, *11*, 341–359.

Tanabe, R., & Fukunaga, A. S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)* (pp. 1658–1665). IEEE.

Tangherloni, A., Rundo, L., & Nobile, M. S. (2017). Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1940–1947). IEEE.

Tian, A.-Q., Chu, S.-C., Pan, J.-S., Cui, H., & Zheng, W.-M. (2020). A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station. *Sustainability*, *12*, 767.

Tian, M., & Gao, X. (2019). Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. *Information Sciences*, *478*, 422–448.

Tighzert, L., Fonlupt, C., Bruneau, O., & Mendil, B. (2017). A new uniform compact evolutionary algorithms. In *2017 5th International Conference on Electrical Engineering-Boumerdes (ICEE-B)* (pp. 1–6). IEEE.

Tighzert, L., Fonlupt, C., & Mendil, B. (2018). A set of new compact firefly algorithms. *Swarm and Evolutionary Computation*, *40*, 92–115.

Tighzert, L., Fonlupt, C., Mendil, B., & Bruneau, O. (2017). New compact music-inspired algorithms. In *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)* (pp. 1–6).

Tighzert, L., & Mendil, B. (2016). Cfo: A new compact swarm intelligent algorithm for global optimization and optimal bipedal robots walking. In *2016 8th International Conference on Modelling, Identification and Control (ICMIC)* (pp. 487–492). IEEE.

Timmerman, K. M. (2012). *A hardware compact genetic algorithm for hover improvement in an insect-scale flapping-wing micro air vehicle*. Master's thesis Wright State University, USA.

Toledo, C. F., da Silva Arantes, M., Oliveira, R. R., & Delbem, A. C. (2013). A hybrid compact genetic algorithm applied to the multi-level capacitated lot sizing problem. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 200–205). ACM.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics* (pp. 196–202). Springer.

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

Xing, H., & Qu, R. (2012). A compact genetic algorithm for the network coding based resource minimization problem. *Applied Intelligence*, *36*, 809–823.

Xingsi, X., Pei-Wei, T., & Jinshui, W. (2017). Using compact memetic algorithm for optimizing ontology alignment. *ICIC Express Letters*, (pp. 53–58).

Xu, C., Huang, H., & Ye, S. (2014). A differential evolution with replacement strategy for real-parameter numerical optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1617–1624). IEEE.

Yang, M., Li, C., Cai, Z., & Guan, J. (2014a). Differential evolution with auto-enhanced population diversity. *IEEE transactions on cybernetics*, *45*, 302–315.

Yang, Z., Li, K., & Guo, Y. (2014b). A new compact teaching-learning-based optimization method. In *International Conference on Intelligent Computing* (pp. 717–726). Springer.

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. *Knowledge-Based Systems*, *159*, 51–62.

Yildizdan, G., & Baykan, Ö. K. (2020). A novel modified bat algorithm hybridizing by differential evolution algorithm. *Expert Systems with Applications*, *141*, 112949.

Zhao, M. (2020). A novel compact cat swarm optimization based on differential method. *Enterprise Information Systems*, *14*, 196–220.

Zhao, M., Pan, J.-S., & Chen, S.-T. (2017). Compact cat swarm optimization algorithm. In *International Conference on Security with Intelligent Computing and Big-data Services* (pp. 33–43). Springer.

Zhou, C., Meng, K., & Qiu, Z. (2002). Compact genetic algorithm mutated by bit. In *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)* (pp. 1836–1839). IEEE volume 3.

Zhu, W., Tang, Y., Fang, J.-A., & Zhang, W. (2013). Adaptive population tuning scheme for differential evolution. *Information Sciences*, *223*, 164–191.

Appendix A. CEC-2014 numerical results against compact algorithms

Table A.12: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against rcGA, cDE, cDE_Exp, DEcDE, cPSO, cTLBO and cFA on CEC-2014 (Liang et al., 2013) in 10 dimensions.

Function	CScDE		rcGA			cDE			cDE_Exp			DEcDE			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
1	1.38E+05	1.20E+05	8.11E+05	2.37E+06	+	**7.37E+04**	**6.06E+04**	-	1.22E+05	1.36E+05	=	7.64E+05	1.35E+06	+	7.94E+05	1.46E+06	+	3.24E+06	2.19E+06	+	5.92E+06	4.38E+06	+
2	3.97E+03	3.51E+03	7.12E+06	3.77E+07	+	4.33E+03	3.46E+03	=	**3.46E+03**	**3.44E+03**	=	6.58E+03	4.46E+03	+	5.94E+03	3.55E+03	+	4.82E+07	2.13E+07	+	7.21E+08	4.34E+08	+
3	6.85E+03	6.41E+03	1.06E+04	7.75E+03	+	**4.83E+03**	**4.54E+03**	=	6.84E+03	6.48E+03	=	6.73E+03	5.97E+03	=	1.28E+04	1.02E+04	+	6.31E+03	3.21E+03	=	1.14E+04	5.94E+03	+
4	1.73E+01	1.81E+01	3.80E+01	1.86E+01	+	**1.66E+01**	**1.80E+01**	=	1.71E+01	1.95E+01	=	2.04E+01	1.90E+01	+	3.56E+01	2.16E+01	+	4.24E+01	1.78E+01	+	4.13E+01	1.21E+01	+
5	2.00E+01	1.46E-03	2.00E+01	4.99E-02	=	1.96E+01	2.80E+00	-	**1.96E+01**	**2.80E+00**	-	2.00E+01	4.52E-03	+	2.00E+01	5.16E-02	=	1.99E+01	1.93E+00	+	2.04E+01	6.08E-01	+
6	3.26E+00	1.28E+00	5.80E+00	1.65E+00	+	**2.83E+00**	**1.12E+00**	=	3.37E+00	1.25E+00	=	2.92E+00	1.16E+00	=	6.35E+00	1.48E+00	+	4.43E+00	1.41E+00	+	7.40E+00	1.21E+00	+
7	**1.69E-01**	**1.02E-01**	2.87E+00	3.44E+00	+	1.69E-01	9.64E-02	=	1.90E-01	1.13E-01	=	2.00E-01	9.43E-02	=	3.96E+00	4.29E+00	+	2.14E+00	5.55E-01	+	6.87E+00	2.07E+00	+
8	2.60E-08	1.86E-07	1.92E+01	9.51E+00	+	2.73E-01	4.48E-01	=	**6.46E-14**	**5.69E-14**	-	1.03E-03	9.36E-04	+	2.20E+01	9.33E+00	+	1.99E+01	8.44E+00	+	5.09E+01	8.98E+00	+
9	1.34E+01	5.38E+00	2.90E+01	1.23E+01	+	**1.15E+01**	**3.38E+00**	=	1.39E+01	4.34E+00	=	1.32E+01	5.11E+00	=	2.85E+01	1.07E+01	+	3.74E+01	7.74E+00	+	5.68E+01	8.94E+00	+
10	3.10E-01	6.96E-01	5.77E+02	2.68E+02	+	2.20E+00	2.64E+00	+	2.74E-01	4.75E-01	=	**1.57E-01**	**4.79E-01**	-	6.36E+02	3.68E+02	+	9.01E+02	1.87E+02	+	1.12E+03	2.66E+02	+
11	5.00E+02	2.34E+02	8.28E+02	2.98E+02	+	**3.73E+02**	**2.03E+02**	-	5.25E+02	2.55E+02	=	5.11E+02	2.31E+02	=	8.81E+02	2.83E+02	+	1.23E+03	2.23E+02	+	1.43E+03	2.35E+02	+
12	1.36E-01	8.53E-02	5.27E-01	3.82E-01	+	1.40E-01	9.06E-02	=	**1.08E-01**	**7.18E-02**	=	1.42E-01	5.99E-02	=	5.26E-01	3.25E-01	+	1.36E+00	1.83E-01	+	1.49E+00	3.15E-01	+
13	3.77E-01	1.49E-01	3.98E-01	1.65E-01	=	3.13E-01	1.09E-01	-	**3.04E-01**	**1.35E-01**	-	3.41E-01	1.10E-01	=	4.24E-01	1.70E-01	=	5.17E-01	1.05E-01	+	7.62E-01	1.91E-01	+
14	4.14E-01	2.78E-01	9.52E-01	1.26E+00	+	3.63E-01	2.79E-01	=	3.85E-01	2.35E-01	=	**2.96E-01**	**1.79E-01**	-	5.83E-01	3.93E-01	+	4.72E-01	2.11E-01	+	8.53E-01	4.03E-01	+
15	1.75E+00	1.34E+00	5.70E+01	1.16E+02	+	1.61E+00	8.04E-01	=	1.76E+00	1.52E+00	=	**1.56E+00**	**8.03E-01**	=	6.87E+01	1.25E+02	+	5.43E+00	8.28E-01	+	3.89E+01	5.55E+01	+
16	2.54E+00	4.74E-01	3.32E+00	3.77E-01	+	2.47E+00	4.49E-01	=	2.60E+00	3.52E-01	=	**2.44E+00**	**4.20E-01**	=	3.37E+00	3.24E-01	+	3.18E+00	2.38E-01	+	3.50E+00	2.43E-01	+
17	1.14E+05	1.72E+05	2.12E+04	2.28E+04	-	1.28E+05	2.24E+05	=	9.14E+04	1.44E+05	=	3.08E+04	5.62E+04	-	5.47E+04	1.01E+05	-	**7.41E+03**	**5.33E+03**	-	4.09E+04	3.50E+04	=
18	1.00E+04	1.03E+04	8.73E+03	1.01E+04	=	1.13E+04	1.27E+04	=	**8.36E+03**	**9.78E+03**	=	9.04E+03	1.11E+04	=	9.90E+03	1.22E+04	=	8.68E+03	8.95E+03	=	3.99E+04	3.64E+04	+
19	1.29E+00	9.69E-01	4.30E+00	1.49E+00	+	**1.06E+00**	**8.03E-01**	=	1.44E+00	1.12E+00	=	1.37E+00	9.07E-01	=	4.77E+00	1.60E+00	+	3.75E+00	6.77E-01	+	5.10E+00	9.74E-01	+
20	8.50E+03	1.07E+04	4.90E+03	6.85E+03	=	5.94E+03	6.52E+03	=	6.78E+03	9.20E+03	=	6.28E+03	9.15E+03	=	1.01E+04	1.19E+04	=	**1.89E+03**	**2.60E+03**	-	5.54E+03	5.25E+03	=
21	1.02E+04	1.30E+04	5.81E+03	7.76E+03	=	4.75E+03	6.59E+03	-	7.74E+03	9.80E+03	=	5.51E+03	8.30E+03	-	7.46E+03	9.39E+03	=	**2.68E+03**	**2.03E+03**	-	9.75E+03	6.90E+03	=
22	7.87E+00	9.11E+00	4.44E+01	3.60E+01	+	**7.59E+00**	**2.07E+01**	=	1.47E+01	3.44E+01	=	9.71E+00	2.63E+01	=	6.00E+01	5.56E+01	+	4.27E+01	1.15E+01	+	7.58E+01	3.55E+01	+
23	**3.29E+02**	**2.87E-13**	3.35E+02	8.91E+00	+	**3.29E+02**	**2.87E-13**	=	**3.29E+02**	**2.87E-13**	=	**3.29E+02**	**3.72E-03**	+	3.32E+02	5.68E+00	+	3.31E+02	9.10E-01	+	3.58E+02	1.01E+01	+
24	1.34E+02	1.16E+01	1.38E+02	1.24E+01	=	**1.28E+02**	**8.74E+00**	-	1.35E+02	1.20E+01	=	1.32E+02	1.12E+01	=	1.40E+02	1.16E+01	+	1.46E+02	8.80E+00	+	1.66E+02	9.12E+00	+
25	1.86E+02	2.59E+01	1.82E+02	3.04E+01	=	1.85E+02	2.85E+01	=	1.82E+02	2.73E+01	=	**1.80E+02**	**2.96E+01**	=	1.90E+02	2.60E+01	=	1.85E+02	2.28E+01	=	1.90E+02	1.60E+01	=
26	1.00E+02	1.30E-01	1.00E+02	1.96E-01	+	**1.00E+02**	**1.47E-01**	-	1.00E+02	1.53E-01	=	1.00E+02	1.14E-01	=	1.00E+02	1.69E-01	+	1.00E+02	1.09E-01	+	1.01E+02	1.77E-01	+
27	2.49E+02	1.98E+02	1.37E+02	1.85E+02	=	3.20E+02	1.49E+02	=	2.30E+02	1.98E+02	=	2.41E+02	1.92E+02	=	1.76E+02	1.98E+02	=	**2.68E+01**	**7.74E+01**	-	1.06E+02	1.68E+02	=
28	4.72E+02	7.77E+01	**4.28E+02**	**6.75E+01**	-	4.46E+02	7.61E+01	-	4.71E+02	9.99E+01	=	4.68E+02	8.43E+01	=	4.34E+02	6.30E+01	-	5.74E+02	6.88E+01	+	4.96E+02	8.81E+01	=
29	5.37E+02	2.50E+02	8.68E+02	5.26E+02	+	**5.20E+02**	**3.42E+02**	=	2.55E+04	1.78E+05	=	6.82E+04	3.38E+05	=	9.96E+02	7.73E+02	+	1.40E+04	1.52E+04	+	2.90E+03	2.71E+03	+
30	9.52E+02	3.04E+02	1.14E+03	5.99E+02	=	9.09E+02	2.88E+02	=	9.47E+02	3.64E+02	=	**8.64E+02**	**2.73E+02**	=	1.56E+03	9.36E+02	+	2.18E+03	7.94E+02	+	1.52E+03	1.06E+03	+

Table A.13: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against rcGA, cDE, cDE_Exp, DEcDE, cPSO, cTLBO and cFA on CEC-2014 (Liang et al., 2013) in 30 dimensions.

Function	CScDE		rcGA			cDE			cDE_Exp			DEcDE			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
1	**3.12E+06**	**1.77E+06**	1.37E+08	1.16E+08	+	1.59E+07	1.17E+07	+	5.80E+06	4.82E+06	+	9.48E+06	5.77E+06	+	8.00E+07	8.41E+07	+	2.20E+08	6.31E+07	+	2.05E+08	9.59E+07	+
2	1.44E+04	1.16E+04	9.71E+09	6.26E+09	+	1.30E+08	3.63E+08	=	**1.06E+04**	**1.19E+04**	=	4.00E+04	2.47E+04	+	1.66E+09	2.73E+09	+	6.70E+09	1.43E+09	+	2.92E+10	6.08E+09	+
3	1.85E+04	1.64E+04	1.17E+05	3.25E+04	+	1.45E+04	1.34E+04	=	2.02E+04	1.78E+04	=	**8.04E+03**	**9.95E+03**	-	7.76E+04	3.66E+04	+	4.32E+04	9.09E+03	+	6.46E+04	1.86E+04	+
4	**8.00E+01**	**4.32E+01**	9.75E+02	5.14E+02	+	1.90E+02	5.91E+01	+	1.19E+02	3.45E+01	+	1.08E+02	4.51E+01	+	5.95E+02	3.80E+02	+	1.27E+03	3.59E+02	+	1.73E+03	1.06E+03	+
5	**2.00E+01**	**1.00E-03**	2.05E+01	1.78E-01	+	2.01E+01	8.43E-02	+	**2.00E+01**	**2.93E-03**	=	2.00E+01	1.82E-02	+	2.03E+01	1.91E-01	+	2.10E+01	5.36E-02	+	2.10E+01	8.90E-02	+
6	1.67E+01	2.92E+00	2.98E+01	4.03E+00	+	1.97E+01	2.94E+00	+	1.73E+01	2.97E+00	=	**1.52E+01**	**3.36E+00**	-	2.94E+01	4.20E+00	+	3.04E+01	2.85E+00	+	3.54E+01	3.11E+00	+
7	**3.69E-02**	**3.25E-02**	7.16E+01	4.25E+01	+	4.49E+00	5.23E+00	+	5.72E-02	7.93E-02	=	1.80E-01	8.36E-02	+	1.74E+01	2.23E+01	+	7.68E+01	1.60E+01	+	2.48E+02	5.99E+01	+
8	**1.96E-02**	**1.39E-01**	1.77E+02	4.14E+01	+	4.38E+01	1.10E+01	+	5.37E+00	2.20E+00	+	7.85E-02	2.43E-01	+	1.46E+02	4.22E+01	+	2.28E+02	2.46E+01	+	3.08E+02	2.61E+01	+
9	8.84E+01	2.05E+01	2.33E+02	4.37E+01	+	1.18E+02	3.19E+01	+	8.98E+01	2.41E+01	=	**8.60E+01**	**1.99E+01**	=	2.00E+02	4.43E+01	+	2.75E+02	2.37E+01	+	3.57E+02	2.61E+01	+
10	**1.46E+00**	**1.10E+00**	4.50E+03	9.73E+02	+	8.75E+02	3.00E+02	+	2.93E+01	3.86E+01	+	1.33E+01	5.84E+00	+	4.08E+03	7.86E+02	+	5.53E+03	3.98E+02	+	6.80E+03	5.53E+02	+
11	2.53E+03	4.41E+02	5.22E+03	8.11E+02	+	2.96E+03	5.51E+02	+	2.72E+03	5.68E+02	=	**2.47E+03**	**4.87E+02**	=	4.56E+03	7.28E+02	+	6.99E+03	3.43E+02	+	7.24E+03	4.88E+02	+
12	**1.43E-01**	**4.83E-02**	1.10E+00	5.52E-01	+	2.77E-01	8.45E-02	+	2.13E-01	8.15E-02	+	1.74E-01	5.83E-02	+	9.46E-01	4.60E-01	+	2.72E+00	2.49E-01	+	2.81E+00	3.86E-01	+
13	5.76E-01	1.49E-01	1.41E+00	9.96E-01	+	7.66E-01	1.47E-01	+	**5.26E-01**	**1.44E-01**	=	6.02E-01	1.43E-01	=	7.60E-01	1.62E-01	+	2.07E+00	5.33E-01	+	3.73E+00	7.38E-01	+
14	3.67E-01	1.37E-01	2.63E+01	2.02E+01	+	1.03E+00	9.74E-01	+	**3.55E-01**	**1.51E-01**	=	3.55E-01	1.39E-01	=	5.99E+00	8.02E+00	+	2.78E+01	5.59E+00	+	4.31E+01	1.88E+01	+
15	**9.42E+00**	**3.95E+00**	1.26E+05	2.53E+05	+	1.91E+02	1.81E+02	+	1.47E+01	8.20E+00	+	1.23E+01	3.54E+00	+	1.71E+04	3.57E+04	+	1.02E+03	1.26E+03	+	2.10E+05	1.52E+05	+
16	**1.03E+01**	**6.62E-01**	1.26E+01	5.12E-01	+	1.14E+01	4.82E-01	+	1.08E+01	6.96E-01	+	1.04E+01	7.38E-01	=	1.26E+01	4.47E-01	+	1.29E+01	1.98E-01	+	1.30E+01	2.47E-01	+
17	1.16E+06	9.32E+05	6.55E+06	6.54E+06	+	8.72E+05	6.64E+05	=	**8.16E+05**	**6.95E+05**	-	1.79E+06	1.07E+06	+	3.60E+06	4.10E+06	+	4.72E+06	2.41E+06	+	6.26E+06	3.31E+06	+
18	**3.71E+03**	**3.49E+03**	3.50E+07	8.82E+07	+	4.74E+03	5.44E+03	=	4.81E+03	5.26E+03	=	4.91E+03	6.84E+03	=	8.11E+06	3.37E+07	+	1.69E+07	9.21E+06	+	1.12E+08	8.46E+07	+
19	**1.66E+01**	**2.16E+01**	1.03E+02	4.46E+01	+	3.76E+01	3.11E+01	+	2.12E+01	2.45E+01	+	2.00E+01	2.60E+01	=	9.26E+01	4.19E+01	+	5.58E+01	2.56E+01	+	1.52E+02	5.19E+01	+
20	2.35E+04	1.51E+04	5.13E+04	3.14E+04	+	2.29E+04	1.43E+04	=	2.50E+04	1.47E+04	=	**5.77E+03**	**9.15E+03**	-	4.90E+04	3.75E+04	+	1.98E+04	1.01E+04	=	3.16E+04	1.68E+04	+
21	6.56E+05	4.50E+05	1.11E+06	1.07E+06	=	**2.79E+05**	**2.64E+05**	-	4.33E+05	4.51E+05	-	7.70E+05	4.74E+05	=	8.91E+05	1.02E+06	=	8.71E+05	7.46E+05	=	1.46E+06	1.09E+06	+
22	6.26E+02	1.95E+02	5.79E+02	2.27E+02	=	5.09E+02	1.86E+02	-	5.95E+02	2.16E+02	=	**5.05E+02**	**1.97E+02**	-	5.86E+02	2.13E+02	=	7.78E+02	1.49E+02	+	8.18E+02	2.07E+02	+
23	**3.15E+02**	**7.50E-03**	4.28E+02	7.99E+01	+	3.19E+02	2.11E+00	+	3.16E+02	1.82E+00	+	3.16E+02	3.07E-01	+	3.67E+02	5.11E+01	+	3.47E+02	8.76E+00	+	4.11E+02	5.22E+01	+
24	2.36E+02	6.80E+00	3.33E+02	3.13E+01	+	2.57E+02	4.94E+00	+	2.37E+02	5.96E+00	+	**2.34E+02**	**5.54E+00**	-	3.13E+02	3.21E+01	+	2.68E+02	3.84E+00	+	3.33E+02	1.72E+01	+
25	2.10E+02	4.84E+00	2.26E+02	1.21E+01	+	**2.09E+02**	**3.13E+00**	=	2.12E+02	5.06E+00	+	2.13E+02	6.86E+00	+	2.22E+02	1.16E+01	+	2.34E+02	7.83E+00	+	2.29E+02	7.74E+00	+
26	1.28E+02	5.82E+01	1.01E+02	9.21E-01	=	1.13E+02	3.27E+01	+	**1.01E+02**	**1.58E-01**	=	1.01E+02	1.29E-01	=	1.01E+02	3.87E-01	=	1.02E+02	4.93E-01	+	1.01E+02	3.26E-01	+
27	7.14E+02	1.97E+02	5.63E+02	1.48E+02	-	7.74E+02	2.08E+02	+	7.22E+02	2.01E+02	=	7.29E+02	1.83E+02	=	**5.49E+02**	**1.81E+02**	-	6.57E+02	1.65E+02	-	6.43E+02	2.43E+02	=
28	1.31E+03	4.07E+02	1.30E+03	3.31E+02	=	**9.84E+02**	**7.69E+01**	-	1.32E+03	3.93E+02	=	1.32E+03	3.85E+02	=	1.15E+03	2.35E+02	=	2.77E+03	4.92E+02	+	1.40E+03	2.74E+02	+
29	1.71E+05	1.21E+06	1.60E+04	4.87E+04	+	**2.10E+03**	**1.43E+03**	+	4.74E+05	1.64E+06	=	1.71E+05	1.21E+06	=	3.61E+03	6.08E+03	+	1.87E+07	1.85E+07	+	1.62E+06	2.37E+06	+
30	3.97E+03	2.91E+03	4.18E+04	3.75E+04	+	5.89E+03	4.93E+03	+	4.74E+03	2.28E+03	+	**3.90E+03**	**1.18E+03**	=	1.70E+04	1.63E+04	+	1.02E+05	4.83E+04	+	6.86E+04	4.47E+04	+

Table A.14: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against rcGA, cDE, cDE_Exp, DEcDE, cPSO, cTLBO and cFA on CEC-2014 (Liang et al., 2013) in 50 dimensions.

Function	CScDE		rcGA			cDE			cDE_Exp			DEcDE			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
1	**3.72E+06**	**1.65E+06**	3.46E+08	1.74E+08	+	1.09E+08	7.85E+07	+	3.45E+07	1.94E+07	+	1.21E+07	4.56E+06	+	1.48E+08	9.23E+07	+	7.04E+08	2.63E+08	+	4.79E+08	1.38E+08	+
2	**6.78E+03**	**8.49E+03**	3.64E+10	1.19E+10	+	6.18E+09	3.15E+09	+	1.47E+07	3.22E+07	+	9.38E+04	7.03E+04	+	1.58E+10	1.09E+10	+	4.43E+10	6.69E+09	+	8.54E+10	1.60E+10	+
3	3.29E+04	1.63E+04	2.23E+05	5.01E+04	+	4.80E+04	1.57E+04	+	3.26E+04	1.48E+04	=	**1.93E+04**	**1.01E+04**	-	1.47E+05	3.99E+04	+	1.34E+05	2.23E+04	+	1.27E+05	2.81E+04	+
4	**9.43E+01**	**3.28E+01**	4.67E+03	3.68E+03	+	1.01E+03	3.01E+02	+	2.35E+02	5.61E+01	+	1.27E+02	3.70E+01	+	1.78E+03	1.15E+03	+	8.35E+03	3.20E+03	+	1.04E+04	5.55E+03	+
5	**2.00E+01**	**7.55E-04**	2.07E+01	1.44E-01	+	2.04E+01	9.94E-02	+	2.00E+01	2.60E-02	+	2.01E+01	2.59E-02	+	2.04E+01	1.63E-01	+	2.12E+01	3.21E-02	+	2.12E+01	3.75E-02	+
6	3.24E+01	3.90E+00	5.75E+01	5.31E+00	+	4.54E+01	3.93E+00	+	3.64E+01	3.53E+00	+	**3.24E+01**	**4.39E+00**	=	5.38E+01	6.09E+00	+	6.13E+01	3.79E+00	+	6.49E+01	3.82E+00	+
7	**2.47E-02**	**3.01E-02**	4.00E+02	1.48E+02	+	7.05E+01	3.67E+01	+	1.30E+00	5.20E-01	+	3.33E-01	1.35E-01	+	1.62E+02	9.32E+01	+	3.94E+02	4.73E+01	+	8.95E+02	1.57E+02	+
8	**5.85E-02**	**2.36E-01**	4.29E+02	8.14E+01	+	1.83E+02	2.38E+01	+	2.26E+01	4.92E+00	+	7.03E-01	7.93E-01	+	3.38E+02	8.56E+01	+	5.11E+02	3.06E+01	+	6.28E+02	4.33E+01	+
9	**1.70E+02**	**4.02E+01**	5.95E+02	1.05E+02	+	3.43E+02	5.64E+01	+	1.94E+02	3.74E+01	+	1.87E+02	4.01E+01	+	4.60E+02	8.37E+01	+	5.77E+02	3.12E+01	+	7.37E+02	5.43E+01	+
10	**1.97E+00**	**1.27E+00**	9.96E+03	9.99E+02	+	3.75E+03	7.65E+02	+	2.41E+02	1.77E+02	+	1.19E+02	1.01E+02	+	8.17E+03	1.24E+03	+	1.13E+04	4.86E+02	+	1.31E+04	7.05E+02	+
11	**5.00E+03**	**7.10E+02**	1.06E+04	1.20E+03	+	7.64E+03	7.77E+02	+	5.27E+03	7.37E+02	=	5.09E+03	7.80E+02	=	9.01E+03	1.11E+03	+	1.33E+04	4.31E+02	+	1.37E+04	4.51E+02	+
12	**1.26E-01**	**3.79E-02**	1.45E+00	5.44E-01	+	4.99E-01	1.30E-01	+	2.53E-01	5.69E-02	+	1.76E-01	5.09E-02	+	1.15E+00	4.52E-01	+	3.55E+00	2.86E-01	+	3.77E+00	4.13E-01	+
13	**6.36E-01**	**1.25E-01**	3.69E+00	8.69E-01	+	7.86E-01	4.21E-01	=	6.49E-01	1.21E-01	=	6.63E-01	1.28E-01	=	1.64E+00	1.20E+00	+	4.41E+00	3.28E-01	+	5.91E+00	6.85E-01	+
14	**4.53E-01**	**2.11E-01**	1.15E+02	4.99E+01	+	2.45E+01	9.95E+00	+	5.40E-01	2.90E-01	=	6.17E-01	3.31E-01	+	4.29E+01	2.27E+01	+	1.01E+02	1.67E+01	+	2.48E+02	3.97E+01	+
15	**1.91E+01**	**6.02E+00**	1.30E+06	1.61E+06	+	2.00E+04	2.28E+04	+	6.24E+01	3.58E+01	+	3.02E+01	9.12E+00	+	2.84E+05	5.33E+05	+	1.54E+05	1.03E+05	+	2.79E+06	1.52E+06	+
16	**1.84E+01**	**8.36E-01**	2.23E+01	5.31E-01	+	2.08E+01	6.66E-01	+	1.96E+01	7.58E-01	+	1.89E+01	8.61E-01	+	2.20E+01	5.12E-01	+	2.26E+01	2.66E-01	+	2.26E+01	3.31E-01	+
17	**1.29E+06**	**7.89E+05**	2.99E+07	2.36E+07	+	3.53E+06	2.47E+06	+	3.37E+06	3.03E+06	+	4.19E+06	2.31E+06	+	1.48E+07	1.08E+07	+	3.49E+07	2.41E+07	+	2.77E+07	1.35E+07	+
18	2.09E+03	1.85E+03	2.40E+08	3.76E+08	+	**1.75E+03**	**1.61E+03**	=	1.92E+03	1.25E+03	=	4.50E+03	4.14E+03	+	2.01E+07	5.81E+07	+	2.71E+08	1.27E+08	+	4.56E+08	2.77E+08	+
19	**2.18E+01**	**1.32E+01**	2.40E+02	1.33E+02	+	9.27E+01	3.31E+01	+	7.21E+01	2.13E+01	+	4.91E+01	2.51E+01	+	1.93E+02	1.20E+02	+	2.05E+02	3.84E+01	+	2.18E+02	4.73E+01	+
20	6.07E+04	2.75E+04	1.71E+05	1.21E+05	+	5.38E+04	2.65E+04	=	6.13E+04	3.25E+04	=	**5.48E+03**	**4.94E+03**	-	1.17E+05	7.28E+04	+	5.62E+04	2.58E+04	=	8.25E+04	5.62E+04	+
21	**1.07E+06**	**6.71E+05**	1.54E+07	9.23E+06	+	1.63E+06	1.79E+06	=	2.05E+06	2.38E+06	+	3.09E+06	1.18E+06	+	7.59E+06	5.43E+06	+	8.19E+06	4.58E+06	+	1.18E+07	6.99E+06	+
22	1.29E+03	2.94E+02	1.53E+03	3.31E+02	+	**1.17E+03**	**3.20E+02**	=	1.22E+03	3.65E+02	=	1.25E+03	3.65E+02	=	1.34E+03	3.77E+02	=	2.38E+03	3.09E+02	+	2.03E+03	2.71E+02	+
23	**3.44E+02**	**4.15E-02**	6.33E+02	1.36E+02	+	3.65E+02	1.44E+01	+	3.53E+02	4.05E+00	+	3.44E+02	1.89E-02	+	4.30E+02	6.32E+01	+	4.68E+02	2.58E+01	+	1.09E+03	1.67E+02	+
24	**2.66E+02**	**5.72E+00**	5.42E+02	5.44E+01	+	3.64E+02	1.38E+01	+	2.80E+02	4.05E+00	+	2.68E+02	6.56E+00	=	4.73E+02	5.27E+01	+	3.71E+02	9.97E+00	+	5.39E+02	3.70E+01	+
25	**2.16E+02**	**5.20E+00**	2.81E+02	2.70E+01	+	2.31E+02	8.45E+00	+	2.24E+02	5.57E+00	+	2.20E+02	5.06E+00	+	2.56E+02	1.86E+01	+	2.91E+02	1.63E+01	+	2.82E+02	2.27E+01	+
26	1.65E+02	7.84E+01	1.39E+02	7.30E+01	=	1.58E+02	7.39E+01	-	1.57E+02	6.88E+01	=	1.52E+02	5.08E+01	=	**1.04E+02**	**1.68E+01**	-	1.28E+02	5.40E+01	=	1.43E+02	6.87E+01	=
27	**1.22E+03**	**1.52E+02**	1.75E+03	1.59E+02	+	1.52E+03	8.30E+01	+	1.32E+03	1.13E+02	+	1.26E+03	1.21E+02	=	1.64E+03	1.35E+02	+	2.09E+03	8.63E+01	+	1.85E+03	1.02E+02	+
28	2.47E+03	7.96E+02	2.75E+03	7.89E+02	+	**1.66E+03**	**3.57E+02**	-	2.61E+03	9.01E+02	=	2.31E+03	7.83E+02	=	2.35E+03	7.22E+02	=	8.22E+03	1.02E+03	+	2.51E+03	7.05E+02	=
29	6.93E+05	4.93E+06	5.47E+06	9.41E+06	+	1.15E+05	2.79E+05	+	6.99E+05	4.93E+06	+	**2.60E+03**	**1.30E+03**	+	3.46E+05	1.89E+06	+	2.57E+08	1.40E+08	+	2.23E+07	1.26E+07	+
30	**1.28E+04**	**2.73E+03**	2.40E+05	1.55E+05	+	3.34E+04	1.63E+04	+	1.84E+04	5.22E+03	+	1.44E+04	2.90E+03	+	6.15E+04	4.27E+04	+	1.26E+06	4.87E+05	+	2.25E+05	1.26E+05	+

Table A.15: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against rcGA, cDE, cDE_Exp, DEcDE, cPSO, cTLBO and cFA on CEC-2014 (Liang et al., 2013) in 100 dimensions.

Function	CScDE		rcGA			cDE			cDE_Exp			DEcDE			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
1	**1.43E+07**	**6.63E+06**	2.25E+09	8.78E+08	+	9.68E+08	3.40E+08	+	2.93E+08	9.23E+07	+	8.47E+07	2.72E+07	+	1.05E+09	5.26E+08	+	2.73E+09	1.11E+09	+	2.28E+09	5.55E+08	+
2	**2.49E+04**	**3.17E+04**	2.29E+11	4.93E+10	+	8.93E+10	1.84E+10	+	5.17E+09	1.99E+09	+	5.88E+05	3.59E+05	+	1.03E+11	4.22E+10	+	1.86E+11	1.67E+10	+	3.11E+11	3.05E+10	+
3	5.71E+04	2.54E+04	5.76E+05	7.08E+04	+	2.16E+05	3.44E+04	+	5.86E+04	1.82E+04	=	**2.61E+04**	**1.48E+04**	-	3.84E+05	7.34E+04	+	3.75E+05	3.81E+04	+	2.81E+05	4.92E+04	+
4	**1.97E+02**	**3.52E+01**	3.56E+04	1.94E+04	+	1.33E+04	4.19E+03	+	1.17E+03	3.49E+02	+	2.61E+02	4.80E+01	+	1.50E+04	8.51E+03	+	4.10E+04	1.09E+04	+	7.18E+04	1.44E+04	+
5	**2.00E+01**	**1.55E-03**	2.10E+01	9.55E-02	+	2.09E+01	6.42E-02	+	2.04E+01	6.52E-02	+	2.02E+01	4.86E-02	+	2.07E+01	1.27E-01	+	2.13E+01	2.78E-02	+	2.14E+01	2.97E-02	+
6	**7.95E+01**	**7.09E+00**	1.34E+02	8.63E+00	+	1.18E+02	5.64E+00	+	9.65E+01	7.43E+00	+	8.52E+01	6.55E+00	+	1.24E+02	8.67E+00	+	1.51E+02	4.02E+00	+	1.47E+02	6.27E+00	+
7	**1.05E-02**	**1.26E-02**	1.78E+03	4.50E+02	+	7.34E+02	1.56E+02	+	4.67E+01	1.72E+01	+	4.50E-01	1.31E-01	+	8.49E+02	3.64E+02	+	1.75E+03	2.27E+02	+	2.84E+03	3.18E+02	+
8	**1.95E-02**	**1.39E-01**	1.30E+03	1.47E+02	+	7.71E+02	5.37E+01	+	1.54E+02	2.38E+01	+	8.35E+00	3.58E+00	+	9.80E+02	1.61E+02	+	1.28E+03	5.11E+01	+	1.51E+03	5.64E+01	+
9	**5.04E+02**	**7.58E+01**	1.52E+03	1.66E+02	+	1.15E+03	1.02E+02	+	7.20E+02	1.05E+02	+	5.73E+02	9.16E+01	+	1.23E+03	1.86E+02	+	1.46E+03	5.82E+01	+	1.73E+03	7.68E+01	+
10	**3.42E+00**	**1.10E+00**	2.49E+04	2.02E+03	+	1.47E+04	1.28E+03	+	3.12E+03	5.82E+02	+	9.50E+02	3.98E+02	+	2.04E+04	2.34E+03	+	2.76E+04	6.16E+02	+	3.05E+04	6.48E+02	+
11	**1.23E+04**	**1.15E+03**	2.56E+04	1.60E+03	+	2.17E+04	1.32E+03	+	1.59E+04	1.11E+03	+	1.24E+04	1.37E+03	=	2.17E+04	2.11E+03	+	3.05E+04	5.30E+02	+	3.07E+04	6.09E+02	+
12	**1.97E-01**	**4.48E-02**	1.94E+00	5.54E-01	+	8.98E-01	1.97E-01	+	5.36E-01	8.83E-02	+	3.14E-01	6.94E-02	+	1.37E+00	5.11E-01	+	4.16E+00	2.31E-01	+	4.32E+00	2.77E-01	+
13	**6.28E-01**	**7.90E-02**	7.35E+00	9.54E-01	+	4.62E+00	3.69E-01	+	6.94E-01	9.11E-02	+	7.11E-01	9.30E-02	+	4.67E+00	1.09E+00	+	7.15E+00	4.71E-01	+	9.26E+00	5.79E-01	+
14	**1.46E-01**	**1.88E-02**	1.39E+02	4.74E+01	+	8.33E+01	2.19E+01	+	1.47E-01	1.84E-02	=	2.41E-01	2.40E-02	+	4.67E+01	2.76E+01	+	1.83E+02	3.77E+01	+	1.45E+02	6.59E+01	+
15	**4.91E+01**	**1.11E+01**	2.47E+07	1.20E+07	+	2.34E+06	1.41E+06	+	6.18E+03	7.29E+03	+	9.25E+01	1.55E+01	+	5.38E+06	7.57E+06	+	5.26E+06	1.98E+06	+	3.45E+07	1.37E+07	+
16	**4.04E+01**	**1.14E+00**	4.61E+01	6.97E-01	+	4.44E+01	7.70E-01	+	4.33E+01	8.06E-01	+	4.25E+01	8.69E-01	+	4.56E+01	8.53E-01	+	4.69E+01	2.63E-01	+	4.68E+01	3.37E-01	+
17	**2.78E+06**	**1.09E+06**	2.33E+08	1.24E+08	+	6.67E+07	3.52E+07	+	4.76E+07	2.23E+07	+	8.59E+06	3.26E+06	+	1.11E+08	5.90E+07	+	2.89E+08	1.04E+08	+	1.92E+08	6.32E+07	+
18	**3.30E+03**	**3.85E+03**	6.64E+09	3.83E+09	+	4.49E+08	4.26E+08	+	2.94E+07	5.49E+07	+	1.57E+05	4.76E+05	+	1.61E+09	1.70E+09	+	3.40E+09	8.54E+08	+	1.47E+10	4.37E+09	+
19	**1.01E+02**	**2.27E+01**	1.10E+03	3.61E+02	+	4.22E+02	8.00E+01	+	2.04E+02	4.23E+01	+	1.18E+02	2.08E+01	+	7.04E+02	2.50E+02	+	1.47E+03	5.69E+02	+	9.77E+02	4.29E+02	+
20	1.07E+05	3.21E+04	1.21E+06	9.61E+05	+	2.23E+05	7.85E+04	+	1.63E+05	4.64E+04	+	**1.75E+04**	**7.71E+03**	-	4.23E+05	3.07E+05	+	2.71E+05	1.25E+05	+	2.87E+05	2.17E+05	+
21	**2.26E+06**	**9.88E+05**	1.06E+08	6.32E+07	+	2.40E+07	1.21E+07	+	1.82E+07	7.32E+06	+	8.09E+06	2.55E+06	+	3.99E+07	2.15E+07	+	1.08E+08	4.02E+07	+	8.16E+07	2.88E+07	+
22	**2.80E+03**	**5.14E+02**	6.16E+03	2.41E+03	+	3.40E+03	5.58E+02	+	3.01E+03	6.12E+02	=	2.83E+03	5.90E+02	=	4.20E+03	1.54E+03	+	5.85E+03	4.30E+02	+	7.43E+03	1.10E+03	+
23	**3.50E+02**	**1.60E+00**	2.19E+03	6.41E+02	+	6.99E+02	9.84E+01	+	5.56E+02	9.23E+01	+	3.50E+02	1.55E+00	+	1.12E+03	3.85E+02	+	8.63E+02	6.54E+01	+	2.86E+03	3.27E+02	+
24	**3.51E+02**	**9.42E+00**	1.18E+03	9.99E+01	+	7.84E+02	4.32E+01	+	4.59E+02	1.38E+01	+	3.63E+02	6.78E+00	+	9.60E+02	1.02E+02	+	7.62E+02	2.70E+01	+	1.14E+03	8.20E+01	+
25	**2.57E+02**	**1.32E+01**	5.43E+02	9.23E+01	+	3.85E+02	3.67E+01	+	3.00E+02	1.69E+01	+	2.61E+02	1.10E+01	=	4.04E+02	5.34E+01	+	6.12E+02	6.04E+01	+	5.24E+02	9.16E+01	+
26	**1.99E+02**	**1.41E+01**	4.36E+02	1.04E+02	+	2.85E+02	4.64E+01	+	2.25E+02	7.18E+00	+	2.00E+02	2.04E+01	+	2.87E+02	7.54E+01	+	4.23E+02	8.22E+01	+	4.69E+02	1.27E+02	+
27	**2.38E+03**	**1.50E+02**	3.64E+03	2.17E+02	+	3.33E+03	1.32E+02	+	2.83E+03	1.61E+02	+	2.53E+03	1.83E+02	+	3.45E+03	2.22E+02	+	4.56E+03	1.26E+02	+	4.00E+03	1.49E+02	+
28	5.60E+03	9.15E+02	7.43E+03	1.26E+03	+	**5.47E+03**	**1.02E+03**	=	6.85E+03	1.46E+03	+	5.71E+03	1.08E+03	=	5.94E+03	1.04E+03	=	2.42E+04	1.49E+03	+	6.94E+03	1.11E+03	+
29	**4.32E+03**	**5.51E+02**	1.06E+08	6.64E+07	+	3.68E+07	1.38E+07	+	5.77E+05	5.10E+05	+	5.47E+03	7.86E+02	+	1.84E+07	2.53E+07	+	1.76E+09	4.22E+08	+	9.50E+07	1.02E+07	+
30	**2.32E+04**	**5.39E+03**	7.09E+06	4.08E+06	+	9.15E+05	4.58E+05	+	2.60E+05	1.39E+05	+	4.22E+04	1.77E+04	+	2.75E+06	3.50E+06	+	1.22E+07	5.35E+06	+	4.46E+06	4.72E+06	+

Appendix B. CEC-2014 numerical results against population-based metaheuristics algorithms

Table B.16: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against b3e3pbest, L-SHADE and RSDE on CEC-2014 (Liang et al., 2013) in 10 dimensions.

Function	CScDE		b3e3pbest			L-SHADE			RSDE		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.38E+05	1.20E+05	3.22E+06	9.21E+06	-	**0.00E+00**	**0.00E+00**	-	1.87E-07	8.33E-07	-
2	3.97E+03	3.51E+03	1.35E+08	4.25E+08	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-
3	6.85E+03	6.41E+03	3.16E+03	7.91E+03	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-
4	1.73E+01	1.81E+01	2.79E+01	5.15E+01	=	2.94E+01	1.26E+01	+	**1.21E+01**	**1.63E+01**	-
5	2.00E+01	1.46E-03	2.02E+01	1.75E-01	+	**1.91E+01**	**2.63E+00**	+	2.00E+01	4.30E-03	-
6	3.26E+00	1.28E+00	3.38E+00	4.03E+00	=	**5.60E-02**	**2.09E-01**	-	3.66E-01	7.00E-01	-
7	1.69E-01	1.02E-01	3.72E+00	1.08E+01	=	**3.83E-02**	**2.74E-02**	-	6.26E-02	3.69E-02	-
8	**2.60E-08**	**1.86E-07**	1.42E+01	2.23E+01	+	2.46E-02	5.64E-02	+	5.80E+00	3.27E+00	+
9	1.34E+01	5.38E+00	2.37E+01	2.15E+01	=	**7.04E+00**	**1.35E+00**	-	1.16E+01	5.47E+00	=
10	**3.10E-01**	**6.96E-01**	3.95E+02	6.18E+02	+	1.26E+01	3.91E+00	+	1.71E+02	1.40E+02	+
11	5.00E+02	2.34E+02	9.49E+02	5.13E+02	+	**3.13E+02**	**1.21E+02**	-	5.14E+02	3.19E+02	=
12	**1.36E-01**	**8.53E-02**	8.67E-01	6.84E-01	+	4.02E-01	6.56E-02	+	2.71E-01	1.85E-01	+
13	3.77E-01	1.49E-01	3.48E-01	3.35E-01	-	**8.53E-02**	**1.75E-02**	-	1.39E-01	3.99E-02	-
14	4.14E-01	2.78E-01	7.73E-01	2.31E+00	-	**1.10E-01**	**3.02E-02**	-	1.58E-01	5.18E-02	-
15	1.75E+00	1.34E+00	7.98E+01	3.25E+02	=	**8.85E-01**	**1.51E-01**	-	9.93E-01	3.75E-01	-
16	2.54E+00	4.74E-01	2.80E+00	6.72E-01	=	**2.16E+00**	**2.51E-01**	-	2.39E+00	4.91E-01	=
17	1.14E+05	1.72E+05	2.79E+04	9.24E+04	-	**2.92E+01**	**1.39E+01**	-	9.72E+01	7.61E+01	-
18	1.00E+04	1.03E+04	3.17E+05	1.56E+06	-	**1.58E+00**	**6.43E-01**	-	3.87E+00	3.66E+00	-
19	1.29E+00	9.69E-01	2.64E+00	3.47E+00	=	**8.66E-01**	**2.74E-01**	-	1.62E+00	6.69E-01	+
20	8.50E+03	1.07E+04	1.11E+04	4.32E+04	-	1.39E+00	3.48E-01	-	**1.09E+00**	**6.94E-01**	-
21	1.02E+04	1.30E+04	1.77E+04	7.43E+04	-	**2.24E+00**	**1.13E+00**	-	1.14E+01	2.76E+01	-
22	**7.87E+00**	**9.11E+00**	7.90E+01	1.43E+02	+	1.10E+01	2.98E+00	+	1.82E+01	7.44E+00	+
23	**3.29E+02**	**2.87E-13**	3.34E+02	1.35E+01	-	**3.29E+02**	**2.87E-13**	-	**3.29E+02**	**2.30E-13**	-
24	1.34E+02	1.16E+01	1.32E+02	2.45E+01	-	**1.12E+02**	**1.90E+00**	-	1.21E+02	7.45E+00	-
25	1.86E+02	2.59E+01	1.77E+02	3.51E+01	-	**1.41E+02**	**3.58E+01**	-	1.49E+02	3.22E+01	-
26	1.00E+02	1.30E-01	1.00E+02	4.46E-01	-	**1.00E+02**	**2.01E-02**	-	1.00E+02	4.12E-02	-
27	2.49E+02	1.98E+02	1.49E+02	1.78E+02	-	**5.84E+01**	**1.34E+02**	-	1.69E+02	1.60E+02	-
28	4.72E+02	7.77E+01	4.31E+02	1.12E+02	-	**3.81E+02**	**3.17E+01**	-	4.06E+02	6.22E+01	-
29	5.37E+02	2.50E+02	1.29E+05	6.27E+05	-	**2.22E+02**	**4.63E-01**	-	3.40E+04	2.41E+05	-
30	9.52E+02	3.04E+02	1.65E+03	2.70E+03	-	**4.65E+02**	**1.34E+01**	-	5.60E+02	1.40E+02	-

Appendix C. Extended numerical results for CScDE on CEC-2017 against population-based algorithms

Appendix D. Details of the tested problems

Appendix D.1. BBOB test functions

The BBOB benchmark consists of 24 functions with different properties in terms of separability, global structure, ill-conditioning, and multi-modality, as summarised in Table D.25. Further information can be found in (Hansen et al., 2009) and (Finck et al., 2010).

$^{(*)}$ The Rosenbrock Function is unimodal in 2D but multimodal in higher dimensionalities (Shang & Qiu, 2006).

Table B.17: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against b3e3pbest, L-SHADE and RSDE on CEC-2014 (Liang et al., 2013) in 30 dimensions.

Function	**CScDE**		**b3e3pbest**			**L-SHADE**			**RSDE**		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	3.12E+06	1.77E+06	7.19E+06	2.50E+07	-	**1.86E-04**	**5.24E-04**	-	1.38E+04	1.06E+04	-
2	1.44E+04	1.16E+04	5.53E+08	2.12E+09	-	**3.07E-09**	**6.72E-09**	-	2.40E-08	1.10E-07	-
3	1.85E+04	1.64E+04	5.57E+03	1.96E+04	-	**0.00E+00**	**0.00E+00**	-	1.59E+00	4.99E+00	-
4	8.00E+01	4.32E+01	1.02E+02	2.43E+02	-	**4.79E+00**	**9.03E+00**	-	2.48E+01	3.26E+01	-
5	**2.00E+01**	**1.00E-03**	2.05E+01	2.06E-01	+	2.07E+01	1.45E-01	+	2.04E+01	1.05E-01	+
6	1.67E+01	2.92E+00	2.24E+01	8.74E+00	+	**4.97E-03**	**7.61E-03**	-	6.47E+00	2.30E+00	-
7	3.69E-02	3.25E-02	4.76E+00	1.79E+01	-	**0.00E+00**	**0.00E+00**	-	2.20E-03	2.98E-03	-
8	**1.96E-02**	**1.39E-01**	5.50E+01	8.22E+01	+	5.10E+01	3.66E+00	+	3.10E+01	9.49E+00	+
9	8.84E+01	2.05E+01	1.15E+02	7.05E+01	=	9.66E+01	9.80E+00	+	**6.36E+01**	**1.81E+01**	-
10	**1.46E+00**	**1.10E+00**	1.67E+03	2.43E+03	+	1.84E+03	2.26E+02	+	1.15E+03	5.40E+02	+
11	**2.53E+03**	**4.41E+02**	4.26E+03	1.55E+03	+	4.80E+03	3.54E+02	+	3.47E+03	7.71E+02	+
12	**1.43E-01**	**4.83E-02**	1.01E+00	7.03E-01	+	1.00E+00	1.44E-01	+	5.25E-01	2.12E-01	+
13	5.76E-01	1.49E-01	3.69E-01	1.55E-01	-	**2.14E-01**	**3.23E-02**	-	3.42E-01	5.51E-02	-
14	3.67E-01	1.37E-01	8.25E-01	3.00E+00	-	2.95E-01	2.88E-02	-	**2.56E-01**	**4.46E-02**	-
15	9.42E+00	3.95E+00	4.22E+02	2.49E+03	=	9.93E+00	7.93E-01	=	**7.74E+00**	**2.67E+00**	-
16	**1.03E+01**	**6.62E-01**	1.14E+01	1.12E+00	+	1.20E+01	4.16E-01	+	1.09E+01	8.33E-01	+
17	1.16E+06	9.32E+05	2.53E+05	1.04E+06	-	**5.87E+02**	**1.79E+02**	-	2.06E+03	9.90E+02	-
18	3.71E+03	3.49E+03	4.02E+06	2.03E+07	-	**3.10E+01**	**4.84E+00**	-	1.29E+02	5.35E+01	-
19	1.66E+01	2.16E+01	1.33E+01	2.58E+01	-	**4.82E+00**	**3.88E-01**	-	8.22E+00	8.20E+00	-
20	2.35E+04	1.51E+04	5.17E+03	1.94E+04	-	**1.94E+01**	**2.77E+00**	-	8.12E+01	5.64E+01	-
21	6.56E+05	4.50E+05	8.13E+04	3.63E+05	-	**4.31E+02**	**1.33E+02**	-	6.32E+02	3.40E+02	-
22	6.26E+02	1.95E+02	4.79E+02	4.02E+02	-	**2.08E+02**	**7.46E+01**	-	2.42E+02	1.24E+02	-
23	3.15E+02	7.50E-03	3.23E+02	2.87E+01	-	**3.15E+02**	**4.02E-13**	-	**3.15E+02**	**2.38E-06**	-
24	2.36E+02	6.80E+00	2.30E+02	1.36E+01	-	**2.24E+02**	**1.05E+00**	-	2.26E+02	3.46E+00	-
25	2.10E+02	4.84E+00	2.06E+02	6.44E+00	-	**2.03E+02**	**4.96E-02**	-	2.03E+02	4.70E-01	-
26	1.28E+02	5.82E+01	1.00E+02	2.67E-01	-	**1.00E+02**	**2.84E-02**	-	1.00E+02	5.88E-02	-
27	7.14E+02	1.97E+02	4.51E+02	2.34E+02	-	**3.00E+02**	**2.74E-04**	-	5.09E+02	1.12E+02	-
28	1.31E+03	4.07E+02	9.73E+02	4.63E+02	-	**8.40E+02**	**1.40E+01**	-	9.84E+02	1.63E+02	-
29	1.71E+05	1.21E+06	3.96E+05	1.66E+06	-	**7.18E+02**	**5.15E+00**	-	2.42E+06	4.72E+06	-
30	3.97E+03	2.91E+03	1.66E+04	5.90E+04	-	**1.30E+03**	**5.83E+02**	-	2.34E+03	1.23E+03	-

Appendix D.2. CEC-2014 test functions

Similar to BBOB, the CEC-2014 includes different types of functions characterised by different properties. More details on these functions can be found in (Liang et al., 2013).

Appendix D.3. Real-world optimisation problems

A total of 22 real-world optimisation problems (both unconstrained and constrained) are considered in the CEC-2011 benchmark. Constrained problems are handled through a static penalty function. The problems, taken from various industrial fields such as chemistry, TLC engineering, and scheduling, are characterised by different dimensionalities and present various optimisation challenges. The main details of the five problems are reported in Table D.27. More details can be found in (Das & Suganthan, 2010).

Table B.18: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against b3e3pbest, L-SHADE and RSDE on CEC-2014 (Liang et al., 2013) in 50 dimensions.

Function	**CScDE**		**b3e3pbest**			**L-SHADE**			**RSDE**		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	3.72E+06	1.65E+06	1.17E+07	3.23E+07	-	**1.83E+04**	**9.86E+03**	-	1.71E+05	1.08E+05	-
2	6.78E+03	8.49E+03	6.70E+08	2.54E+09	=	**2.49E-01**	**2.29E-01**	-	4.40E+03	6.92E+03	-
3	3.29E+04	1.63E+04	6.90E+03	2.43E+04	-	**0.00E+00**	**0.00E+00**	-	5.05E+01	9.26E+01	-
4	9.43E+01	3.28E+01	1.55E+02	2.71E+02	-	9.39E+01	4.91E+00	=	**9.11E+01**	**4.21E+01**	=
5	**2.00E+01**	**7.55E-04**	2.06E+01	2.11E-01	+	2.10E+01	1.32E-01	+	2.06E+01	9.62E-02	+
6	3.24E+01	3.90E+00	4.22E+01	1.34E+01	+	**2.66E-01**	**5.23E-01**	-	2.00E+01	4.44E+00	-
7	2.47E-02	3.01E-02	5.34E+00	1.93E+01	=	**0.00E+00**	**0.00E+00**	-	6.76E-03	8.95E-03	-
8	**5.85E-02**	**2.36E-01**	1.04E+02	1.53E+02	+	1.54E+02	8.29E+00	+	8.87E+01	2.69E+01	+
9	1.70E+02	4.02E+01	2.31E+02	1.15E+02	=	2.21E+02	1.24E+01	+	**1.49E+02**	**3.43E+01**	-
10	**1.97E+00**	**1.27E+00**	3.08E+03	4.40E+03	+	6.00E+03	3.03E+02	+	3.99E+03	1.40E+03	+
11	**5.00E+03**	**7.10E+02**	8.06E+03	2.55E+03	+	1.15E+04	8.68E+02	+	7.16E+03	9.95E+02	+
12	**1.26E-01**	**3.79E-02**	1.12E+00	7.94E-01	+	2.25E+00	6.68E-01	+	6.17E-01	2.35E-01	+
13	6.36E-01	1.25E-01	4.73E-01	1.31E-01	-	**2.95E-01**	**4.00E-02**	-	4.73E-01	5.60E-02	-
14	4.53E-01	2.11E-01	8.87E-01	3.30E+00	-	3.68E-01	2.79E-02	=	**3.11E-01**	**7.35E-02**	-
15	**1.91E+01**	**6.02E+00**	2.77E+02	1.17E+03	=	2.31E+01	1.19E+00	+	1.93E+01	7.46E+00	=
16	**1.84E+01**	**8.36E-01**	2.01E+01	1.37E+00	+	2.20E+01	3.95E-01	+	1.99E+01	9.78E-01	+
17	1.29E+06	7.89E+05	1.90E+05	7.73E+05	-	**1.40E+03**	**5.12E+02**	-	9.01E+03	4.39E+03	-
18	2.09E+03	1.85E+03	2.98E+06	1.54E+07	-	**9.75E+01**	**1.38E+01**	-	8.64E+02	1.01E+03	-
19	2.18E+01	1.32E+01	3.08E+01	2.37E+01	=	**1.22E+01**	**3.72E-01**	-	1.71E+01	4.48E+00	-
20	6.07E+04	2.75E+04	4.09E+03	1.49E+04	-	**6.74E+01**	**8.74E+00**	-	4.50E+02	3.58E+02	-
21	1.07E+06	6.71E+05	1.49E+05	6.83E+05	-	**8.02E+02**	**2.67E+02**	-	3.05E+03	1.64E+03	-
22	1.29E+03	2.94E+02	1.27E+03	6.59E+02	=	8.10E+02	1.44E+02	-	**6.06E+02**	**2.69E+02**	-
23	3.44E+02	4.15E-02	3.50E+02	2.23E+01	+	**3.44E+02**	**4.44E-13**	+	**3.44E+02**	**1.81E-05**	=
24	**2.66E+02**	**5.72E+00**	2.92E+02	1.51E+01	+	2.75E+02	6.62E-01	+	2.78E+02	2.91E+00	+
25	2.16E+02	5.20E+00	2.13E+02	8.00E+00	-	**2.05E+02**	**3.65E-01**	-	2.08E+02	1.62E+00	-
26	1.65E+02	7.84E+01	**1.01E+02**	**5.01E-01**	-	1.02E+02	1.40E+01	-	1.24E+02	7.03E+01	-
27	1.22E+03	1.52E+02	7.32E+02	4.14E+02	-	**3.33E+02**	**3.03E+01**	-	8.85E+02	1.14E+02	-
28	2.47E+03	7.96E+02	1.59E+03	1.18E+03	-	**1.11E+03**	**2.91E+01**	-	1.83E+03	5.10E+02	-
29	6.93E+05	4.93E+06	2.16E+06	1.14E+07	-	**8.36E+02**	**3.93E+01**	-	1.29E+07	2.77E+07	-
30	1.28E+04	2.73E+03	3.01E+04	5.59E+04	+	**8.67E+03**	**4.13E+02**	-	1.26E+04	2.69E+03	=

Table B.19: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against b3e3pbest, L-SHADE and RSDE on CEC-2014 (Liang et al., 2013) in 100 dimensions.

Function	**CScDE**		**b3e3pbest**			**L-SHADE**			**RSDE**		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.43E+07	6.63E+06	3.48E+07	6.98E+07	-	**8.39E+05**	**1.81E+05**	-	1.99E+06	5.86E+05	-
2	2.49E+04	3.17E+04	1.47E+09	5.46E+09	=	**2.47E+03**	**1.48E+03**	-	1.50E+04	2.17E+04	=
3	5.71E+04	2.54E+04	2.12E+04	3.62E+04	-	**7.52E-04**	**4.89E-04**	-	2.04E+02	2.39E+02	-
4	1.97E+02	3.52E+01	3.76E+02	5.15E+02	=	**1.92E+02**	**2.49E+01**	=	2.22E+02	4.70E+01	+
5	**2.00E+01**	**1.55E-03**	2.09E+01	1.59E-01	+	2.13E+01	6.54E-02	+	2.09E+01	8.38E-02	+
6	7.95E+01	7.09E+00	9.69E+01	2.62E+01	+	**8.70E+00**	**2.33E+00**	-	6.58E+01	8.33E+00	-
7	1.05E-02	1.26E-02	1.56E+01	5.35E+01	+	**0.00E+00**	**0.00E+00**	-	1.37E-02	5.22E-02	=
8	**1.95E-02**	**1.39E-01**	2.38E+02	3.48E+02	+	4.92E+02	1.68E+01	+	2.60E+02	3.91E+01	+
9	5.04E+02	7.58E+01	5.77E+02	2.42E+02	=	5.81E+02	2.31E+01	+	**3.77E+02**	**7.54E+01**	-
10	**3.42E+00**	**1.10E+00**	6.98E+03	9.88E+03	+	1.93E+04	4.96E+02	+	1.43E+04	3.32E+03	+
11	**1.23E+04**	**1.15E+03**	1.90E+04	5.05E+03	+	2.92E+04	7.69E+02	+	1.70E+04	2.86E+03	+
12	**1.97E-01**	**4.48E-02**	1.38E+00	7.83E-01	+	3.83E+00	4.03E-01	+	8.70E-01	2.71E-01	+
13	6.28E-01	7.90E-02	5.10E-01	8.57E-02	-	**2.14E-01**	**3.23E-02**	-	5.70E-01	4.07E-02	-
14	**1.46E-01**	**1.88E-02**	1.97E+00	7.31E+00	+	1.79E-01	2.02E-02	+	2.17E-01	1.83E-02	+
15	**4.91E+01**	**1.11E+01**	1.45E+03	6.47E+03	+	5.85E+01	1.92E+00	+	1.43E+02	7.05E+01	+
16	**4.04E+01**	**1.14E+00**	4.25E+01	1.90E+00	+	4.65E+01	2.54E-01	+	4.32E+01	1.26E+00	+
17	2.78E+06	1.09E+06	2.49E+06	6.12E+06	-	**5.01E+03**	**8.09E+02**	-	1.36E+05	5.83E+04	-
18	3.30E+03	3.85E+03	7.83E+05	3.34E+06	+	**4.32E+02**	**2.83E+01**	-	2.20E+03	1.86E+03	=
19	1.01E+02	2.27E+01	1.22E+02	4.75E+01	+	9.69E+01	2.02E+00	-	**9.03E+01**	**2.51E+01**	=
20	1.07E+05	3.21E+04	1.20E+04	2.13E+04	-	**2.10E+02**	**5.08E+01**	-	1.07E+03	4.02E+02	-
21	2.26E+06	9.88E+05	4.36E+05	6.76E+05	-	**2.68E+03**	**5.22E+02**	-	5.39E+04	2.53E+04	-
22	2.80E+03	5.14E+02	3.31E+03	1.40E+03	=	3.14E+03	2.46E+02	+	**1.86E+03**	**5.01E+02**	-
23	3.50E+02	1.60E+00	3.59E+02	3.77E+01	-	**3.48E+02**	**7.56E-07**	-	3.48E+02	5.41E-03	-
24	**3.51E+02**	**9.42E+00**	4.20E+02	2.16E+01	+	3.94E+02	2.87E+00	+	4.13E+02	7.17E+00	+
25	2.57E+02	1.32E+01	2.76E+02	8.33E+00	+	**2.00E+02**	**3.42E-07**	-	2.64E+02	1.15E+01	+
26	**1.99E+02**	**1.41E+01**	2.01E+02	1.44E+00	-	2.00E+02	1.19E-02	-	2.03E+02	5.08E+01	-
27	2.38E+03	1.50E+02	1.88E+03	3.58E+02	-	**3.77E+02**	**3.28E+01**	-	2.25E+03	1.88E+02	-
28	5.60E+03	9.15E+02	3.59E+03	2.46E+03	-	**2.31E+03**	**4.56E+01**	-	4.81E+03	8.30E+02	-
29	4.32E+03	5.51E+02	7.12E+05	4.13E+06	-	**1.96E+03**	**2.28E+02**	-	1.04E+08	9.59E+07	=
30	2.32E+04	5.39E+03	3.87E+04	1.02E+05	-	**8.59E+03**	**1.04E+03**	-	1.41E+04	2.72E+03	-

Table C.20: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 (Awad et al., November 2016) in 10 dimensions.

Function	CScDE		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	2.95E+03	2.91E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.54E-09	7.33E-09	-	2.41E+02	2.02E+02	-	2.10E+03	2.50E+03	=
2	1.65E+03	3.05E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	2.16E-01	1.54E+00	-	3.92E-02	1.96E-01	-
3	2.88E+00	1.29E+01	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	3.14E-04	4.29E-04	-	1.94E+01	1.10E+01	+
4	5.93E+00	1.23E+01	1.24E-03	3.07E-03	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.90E+00	1.46E+00	-	3.97E+00	1.21E+00	=
5	1.36E+01	4.83E+00	**1.83E+00**	**1.60E+00**	-	9.31E+00	2.39E+00	-	5.39E+00	1.18E+00	-	2.13E+00	2.03E+00	-	1.81E+01	5.10E+00	+	1.63E+01	5.29E+00	+
6	5.79E-02	6.94E-02	7.58E-08	3.64E-08	-	3.57E-05	1.53E-05	-	**0.00E+00**	**0.00E+00**	-	1.34E-04	7.12E-04	-	9.18E-01	1.87E+00	+	8.40E-08	4.44E-07	-
7	2.61E+01	9.35E+00	1.16E+01	6.16E-01	-	2.27E+01	2.89E+00	=	1.43E+01	1.29E+00	-	**1.11E+01**	**2.61E+00**	-	1.78E+01	2.36E+00	-	3.22E+01	3.79E+00	+
8	1.67E+01	6.36E+00	**1.68E+00**	**1.16E+00**	-	1.04E+01	2.56E+00	-	4.26E+00	1.34E+00	-	3.22E+00	3.29E+00	-	9.95E+00	2.37E+00	-	2.01E+01	4.38E+00	+
9	1.38E+01	2.30E+01	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	2.80E-04	1.49E-03	-	8.91E-03	6.36E-02	-
10	5.79E+02	1.84E+02	3.19E+02	1.19E+02	-	7.01E+02	1.78E+02	+	**2.86E+02**	**1.01E+02**	-	5.39E+02	2.12E+02	=	5.03E+02	1.55E+02	=	1.14E+03	2.07E+02	+
11	1.11E+01	6.74E+00	1.67E+00	1.02E+00	-	3.22E+00	5.52E-01	-	**1.19E-06**	**8.51E-06**	-	9.03E-01	8.58E-01	-	1.73E+01	5.53E+00	+	4.49E+00	1.49E+00	-
12	1.62E+05	4.04E+05	9.82E+01	7.08E+01	-	**2.44E+01**	**1.96E+01**	-	1.19E+02	7.16E+01	-	2.19E+02	1.50E+02	-	4.76E+03	2.58E+03	-	1.38E+05	1.60E+05	+
13	1.09E+04	1.10E+04	**4.62E+00**	**2.97E+00**	-	7.74E+00	1.88E+00	-	4.94E+00	2.53E+00	-	8.80E+00	2.65E+01	-	1.46E+03	1.41E+03	-	3.20E+03	2.76E+03	-
14	4.47E+03	6.59E+03	3.67E+00	3.46E+00	-	7.18E+00	1.89E+00	-	**3.32E+00**	**7.28E+00**	-	2.52E+01	1.23E+01	-	6.29E+01	6.84E+01	-	9.46E+01	3.37E+01	-
15	6.85E+03	7.85E+03	**2.69E-01**	**1.27E-01**	-	6.38E-01	1.94E-01	-	4.08E-01	2.29E-01	-	3.14E+01	4.19E+01	-	1.87E+02	1.86E+02	-	2.14E+02	7.54E+01	-
16	1.42E+02	1.12E+02	3.42E+00	6.35E-01	-	5.41E+00	1.44E+00	-	**1.41E+00**	**8.39E-01**	-	1.49E+02	1.32E+02	=	8.36E+01	7.26E+01	-	1.23E+01	2.40E+01	-
17	4.29E+01	4.70E+01	1.21E+01	3.80E+00	-	3.44E+01	3.72E+00	+	**1.15E+01**	**6.34E+00**	-	6.61E+01	3.86E+01	+	2.47E+01	7.47E+00	=	4.65E+01	9.62E+00	+
18	1.34E+04	1.14E+04	1.41E+00	4.24E+00	-	**4.84E-01**	**2.04E-01**	-	5.96E+00	9.00E+00	-	4.43E+01	5.10E+01	-	2.26E+03	2.19E+03	-	9.41E+03	6.52E+03	=
19	5.36E+03	6.98E+03	**2.23E-01**	**1.37E-01**	-	1.19E+00	3.34E-01	-	2.35E-01	3.79E-01	-	3.63E+00	3.65E+00	-	1.48E+02	2.38E+02	-	1.24E+02	1.08E+02	-
20	3.16E+00	4.64E+00	**1.74E+00**	**2.03E+00**	=	2.15E+01	3.97E+00	+	2.71E+00	6.51E+00	-	1.23E+02	7.83E+01	+	2.98E+01	1.09E+01	+	2.26E+01	9.74E+00	+
21	1.92E+02	5.33E+01	1.39E+02	3.97E+01	-	1.40E+02	5.13E+01	-	**1.02E+02**	**1.01E+01**	-	1.55E+02	5.24E+01	-	1.05E+02	2.15E+01	-	1.48E+02	5.46E+01	-
22	1.32E+02	1.65E+02	9.85E+01	1.10E+01	-	1.00E+02	1.80E-01	-	1.00E+02	1.13E-01	-	9.93E+01	5.57E+00	-	9.68E+01	1.68E+01	-	**9.37E+01**	**2.22E+01**	-
23	3.21E+02	1.01E+01	3.04E+02	2.01E+00	-	3.07E+02	2.15E+00	-	3.06E+02	1.24E+00	-	**2.87E+02**	**5.52E+01**	-	3.42E+02	1.05E+01	+	3.11E+02	6.41E+00	-
24	3.28E+02	8.77E+01	**2.14E+02**	**8.48E+01**	-	3.28E+02	3.57E+01	-	2.90E+02	8.55E+01	-	2.33E+02	1.10E+02	-	2.27E+02	1.35E+02	-	3.14E+02	6.72E+01	-
25	4.24E+02	2.45E+01	4.13E+02	2.10E+01	-	4.06E+02	1.75E+01	-	4.28E+02	2.16E+01	-	4.07E+02	6.65E+01	-	**4.04E+02**	**1.45E+01**	-	4.26E+02	2.25E+01	=
26	4.41E+02	3.30E+02	**2.66E+02**	**4.67E+01**	-	3.00E+02	0.00E+00	-	3.00E+02	0.00E+00	-	2.85E+02	1.45E+02	-	2.67E+02	7.66E+01	-	3.02E+02	4.66E+01	-
27	4.05E+02	1.53E+01	3.92E+02	2.40E+00	-	**3.89E+02**	**2.22E-01**	-	3.90E+02	9.32E-01	-	3.95E+02	3.98E+00	-	4.27E+02	1.35E+01	+	3.93E+02	3.35E+00	-
28	4.47E+02	1.37E+02	3.22E+02	9.83E+01	-	3.43E+02	1.00E+02	-	3.17E+02	6.97E+01	-	4.18E+02	1.71E+02	=	**2.94E+02**	**4.21E+01**	-	4.49E+02	1.56E+02	=
29	3.08E+02	5.67E+01	**2.45E+02**	**5.82E+00**	-	2.57E+02	6.87E+00	-	2.48E+02	6.52E+00	-	2.87E+02	5.30E+01	-	2.78E+02	1.36E+01	-	2.87E+02	1.63E+01	=
30	3.10E+05	4.54E+05	4.09E+02	1.82E+01	-	**3.95E+02**	**1.01E-01**	-	4.30E+02	2.59E+01	-	2.29E+03	1.04E+04	-	7.09E+03	3.12E+03	-	2.98E+05	5.08E+05	-

Table C.21: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 (Awad et al., November 2016) in 30 dimensions.

Function	CScDE		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	5.29E+03	6.23E+03	1.25E-04	8.32E-05	-	3.68E-06	1.95E-06	-	**0.00E+00**	**0.00E+00**	-	6.27E-08	1.93E-08	-	7.55E+02	6.11E+02	-	3.66E+03	3.76E+03	=
2	6.78E+16	4.69E+17	4.45E+00	1.40E+01	-	**1.72E-08**	**6.48E-08**	-	1.96E-02	1.40E-01	-	1.17E+02	8.35E+02	-	9.50E+06	1.71E+07	-	8.52E+16	5.81E+17	=
3	3.07E+04	1.65E+04	3.59E+01	2.40E+02	-	4.03E-08	1.98E-08	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	4.08E+02	2.03E+02	-	2.08E+04	4.64E+03	-
4	7.27E+01	2.54E+01	6.93E+01	1.17E+01	-	5.87E+01	7.78E-01	-	5.91E+01	3.45E+00	-	**5.53E+01**	**1.65E+01**	-	5.88E+01	3.40E+01	-	9.49E+01	2.35E+01	+
5	8.39E+01	2.35E+01	1.17E+01	3.96E+00	-	1.25E+02	1.07E+01	+	3.03E+01	4.32E+01	-	**4.00E+00**	**1.94E+00**	-	1.12E+02	1.33E+01	+	7.57E+01	4.35E+01	=
6	4.91E-02	4.48E-02	4.60E-05	1.42E-05	-	2.92E-04	5.66E-05	-	**1.31E-07**	**9.11E-08**	-	5.40E-05	3.78E-04	-	2.16E+01	3.97E+00	+	4.87E-01	4.24E-01	+
7	1.24E+02	2.51E+01	3.94E+01	1.23E+01	-	1.70E+02	1.33E+01	+	4.50E+01	2.47E+01	-	**3.52E+01**	**1.81E+00**	-	1.35E+02	1.62E+01	+	1.89E+02	2.09E+01	+
8	1.00E+02	2.47E+01	2.39E+01	3.50E+01	-	1.25E+02	1.23E+01	+	6.99E+00	1.93E+01	-	**4.35E+00**	**2.66E+00**	-	8.10E+01	1.04E+01	-	7.14E+01	4.37E+01	-
9	1.16E+03	1.23E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	1.38E+03	2.72E+02	+	3.46E+01	2.73E+01	-
10	2.59E+03	4.52E+02	4.40E+03	7.91E+02	+	6.19E+03	3.00E+02	+	5.28E+03	8.28E+02	+	**1.87E+03**	**7.56E+02**	-	3.13E+03	3.46E+02	+	6.99E+03	2.62E+02	+
11	5.44E+01	3.00E+01	4.40E+01	1.78E+01	=	3.19E+01	1.35E+01	-	**1.78E+01**	**2.48E+01**	-	6.32E+01	3.96E+01	=	8.83E+01	2.12E+01	+	8.99E+01	4.19E+01	+
12	1.09E+06	9.37E+05	2.53E+03	1.27E+03	-	**5.78E+02**	**3.04E+02**	-	6.15E+02	3.13E+02	-	1.38E+03	3.62E+02	-	9.51E+04	5.13E+04	-	1.04E+05	1.70E+05	-
13	1.86E+04	2.08E+04	8.02E+01	1.96E+01	-	6.68E+01	8.38E+00	-	**2.90E+01**	**2.42E+01**	-	9.97E+02	9.09E+02	-	3.21E+03	2.88E+03	-	2.06E+04	1.78E+04	=
14	3.83E+05	3.99E+05	6.36E+01	3.95E+00	-	4.86E+01	5.59E+00	-	**2.61E+01**	**8.63E+00**	-	1.74E+02	4.94E+01	-	3.48E+03	2.63E+03	-	1.13E+04	8.70E+03	-
15	1.14E+04	1.03E+04	3.75E+01	5.32E+00	-	2.36E+01	3.04E+00	-	**6.51E+00**	**2.41E+00**	-	2.75E+02	1.57E+02	-	2.13E+03	1.63E+03	-	2.90E+04	2.57E+04	+
16	1.15E+03	2.82E+02	9.04E+02	2.05E+02	-	8.20E+02	1.79E+02	-	**4.75E+02**	**3.40E+02**	-	5.83E+02	2.48E+02	-	8.46E+02	1.53E+02	-	7.48E+02	4.33E+02	-
17	4.76E+02	2.04E+02	1.78E+02	3.36E+01	-	2.14E+02	2.57E+01	-	1.88E+02	4.84E+01	-	2.48E+02	1.60E+02	-	3.31E+02	1.13E+02	-	**1.52E+02**	**7.04E+01**	-
18	1.67E+06	1.67E+06	4.19E+01	4.31E+00	-	3.05E+01	1.84E+00	-	**2.39E+01**	**1.92E+00**	-	2.20E+02	1.23E+02	-	8.77E+04	3.31E+04	-	5.31E+05	2.20E+05	-
19	1.01E+04	1.15E+04	2.26E+01	2.14E+00	-	2.41E+01	1.70E+00	-	**1.08E+01**	**2.94E+00**	-	1.91E+02	7.37E+01	-	1.71E+03	1.69E+03	-	1.23E+04	1.13E+04	=
20	5.75E+02	1.80E+02	2.64E+02	5.36E+01	-	3.12E+02	4.74E+01	-	**1.56E+02**	**1.01E+02**	-	4.97E+02	2.03E+02	-	3.58E+02	1.17E+02	-	3.33E+02	1.37E+02	-
21	3.04E+02	2.62E+01	2.16E+02	1.59E+01	-	3.19E+02	9.95E+00	+	2.83E+02	5.58E+01	=	**2.13E+02**	**4.75E+00**	-	3.05E+02	3.30E+01	=	2.65E+02	4.14E+01	-
22	3.03E+03	1.01E+03	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	**1.00E+02**	**0.00E+00**	-	1.09E+03	1.12E+03	-	**1.00E+02**	**2.36E-03**	-	1.01E+02	1.94E+00	-
23	4.46E+02	2.33E+01	3.80E+02	2.38E+01	-	4.67E+02	1.08E+01	+	4.55E+02	1.47E+01	=	**3.47E+02**	**5.02E+01**	-	6.81E+02	3.79E+01	+	3.97E+02	1.62E+01	-
24	6.32E+02	1.09E+02	4.54E+02	3.48E+01	-	5.31E+02	8.43E+00	-	5.41E+02	2.90E+01	-	**4.23E+02**	**3.24E+00**	-	7.39E+02	4.57E+01	+	4.72E+02	2.00E+01	-
25	3.99E+02	1.69E+01	3.87E+02	2.14E-02	-	3.87E+02	7.72E-03	-	3.87E+02	1.06E-02	-	3.87E+02	1.49E-02	-	**3.86E+02**	**1.77E+00**	-	4.02E+02	1.76E+01	=
26	2.12E+03	5.05E+02	7.75E+02	3.46E+02	-	2.00E+03	9.68E+01	-	1.09E+03	3.17E+02	-	**5.09E+02**	**2.71E+02**	-	2.04E+03	1.73E+03	=	1.43E+03	4.62E+02	-
27	5.39E+02	1.76E+01	5.02E+02	4.03E+00	-	**4.99E+02**	**7.99E+00**	-	5.05E+02	6.37E+00	-	5.20E+02	1.32E+01	-	7.08E+02	5.41E+01	+	5.32E+02	2.07E+01	-
28	3.97E+02	5.05E+01	3.61E+02	4.43E+01	-	3.13E+02	3.36E+01	-	3.14E+02	3.52E+01	-	**3.13E+02**	**3.48E+01**	-	3.91E+02	1.77E+01	-	4.35E+02	2.69E+01	+
29	9.66E+02	2.49E+02	7.33E+02	8.42E+01	-	7.51E+02	3.73E+01	-	**4.89E+02**	**7.49E+01**	-	5.68E+02	1.54E+02	-	7.81E+02	1.21E+02	-	6.26E+02	9.19E+01	-
30	9.15E+03	3.54E+03	2.04E+03	6.20E+01	-	**1.98E+03**	**1.88E+01**	-	2.01E+03	6.24E+01	-	3.07E+03	1.93E+03	-	5.69E+03	1.94E+03	-	3.65E+04	3.88E+04	+

Table C.22: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 (Awad et al., November 2016) in 50 dimensions.

Function	CScDE		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	7.80E+03	9.38E+03	7.30E+02	4.56E+02	-	2.44E-01	2.97E-01	-	2.57E-02	2.86E-02	-	**1.23E-07**	**4.41E-08**	-	4.97E+03	2.50E+03	=	1.02E+06	3.03E+06	+
2	1.36E+49	9.72E+49	1.25E+04	4.86E+04	-	3.77E+04	1.96E+05	-	**1.87E+02**	**5.84E+02**	-	2.77E+05	1.98E+06	-	1.25E+18	4.36E+18	-	1.57E+39	1.01E+40	+
3	8.04E+04	2.35E+04	3.51E+03	1.04E+04	-	2.18E-04	1.56E-04	-	3.57E-08	3.23E-08	-	**0.00E+00**	**0.00E+00**	-	1.54E+04	3.23E+03	-	7.02E+04	9.11E+03	-
4	1.02E+02	5.32E+01	6.83E+01	4.60E+01	-	8.19E+01	5.67E+01	-	7.53E+01	5.51E+01	-	**3.82E+01**	**4.42E+01**	-	1.11E+02	4.11E+01	=	1.98E+02	4.63E+01	+
5	1.84E+02	4.14E+01	1.11E+02	1.04E+02	-	2.72E+02	1.16E+01	+	**5.99E+00**	**2.02E+00**	-	7.47E+00	2.74E+00	-	2.01E+02	1.37E+01	+	1.01E+02	1.80E+01	-
6	4.49E-02	2.98E-02	2.24E-04	2.35E-05	-	9.82E-05	2.94E-05	-	1.46E-05	6.24E-06	-	**1.89E-07**	**1.72E-07**	-	3.37E+01	3.92E+00	+	4.52E+00	1.70E+00	+
7	2.29E+02	3.89E+01	1.22E+02	5.00E+01	-	3.39E+02	1.56E+01	+	6.08E+01	2.56E+01	-	**5.93E+01**	**2.09E+00**	-	2.78E+02	3.42E+01	+	2.60E+02	1.04E+02	=
8	1.75E+02	4.13E+01	1.46E+02	1.18E+02	=	2.71E+02	1.76E+01	+	**5.81E+00**	**1.88E+00**	-	7.94E+00	2.54E+00	-	1.99E+02	1.52E+01	+	9.73E+01	1.54E+01	-
9	3.18E+03	2.30E+03	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	**0.00E+00**	**0.00E+00**	-	6.13E+03	7.42E+02	+	1.73E+03	1.19E+03	-
10	4.56E+03	6.94E+02	7.70E+03	9.00E+02	+	1.21E+04	3.37E+02	+	9.37E+03	3.57E+03	+	**3.00E+03**	**9.51E+02**	-	5.20E+03	5.52E+02	+	1.29E+04	4.00E+02	+
11	1.37E+02	3.61E+01	1.32E+02	9.01E+00	=	9.67E+01	8.37E+00	-	**3.24E+01**	**4.63E+00**	-	2.13E+02	5.70E+01	+	1.29E+02	1.43E+01	=	1.74E+02	4.91E+01	+
12	2.61E+06	1.93E+06	2.97E+04	1.34E+04	-	1.75E+03	5.18E+02	-	**1.60E+03**	**4.19E+02**	-	3.89E+06	2.76E+07	-	1.01E+06	3.45E+05	-	1.30E+06	1.05E+06	-
13	9.27E+03	9.67E+03	3.04E+02	3.78E+01	-	1.94E+02	3.40E+01	-	**1.57E+02**	**4.53E+01**	-	2.32E+03	1.60E+03	-	9.01E+02	5.62E+02	-	8.40E+03	5.19E+03	=
14	6.11E+05	5.06E+05	1.36E+02	9.33E+00	-	1.01E+02	7.72E+00	-	**3.11E+01**	**4.86E+00**	-	2.93E+02	8.35E+01	-	2.60E+04	1.34E+04	-	1.25E+05	6.54E+04	-
15	9.18E+03	6.09E+03	1.27E+02	1.27E+01	-	8.36E+01	8.41E+00	-	**3.08E+01**	**5.59E+00**	-	8.25E+02	2.08E+02	-	1.20E+03	7.81E+02	-	6.97E+03	6.03E+03	=
16	2.13E+03	4.54E+02	1.93E+03	3.90E+02	-	1.96E+03	1.98E+02	-	1.02E+03	6.77E+02	-	**9.44E+02**	**3.40E+02**	-	1.25E+03	2.31E+02	-	9.78E+02	3.64E+02	-
17	1.46E+03	2.67E+02	1.47E+03	2.16E+02	=	1.32E+03	1.29E+02	-	**6.34E+02**	**4.50E+02**	-	8.46E+02	2.34E+02	-	1.03E+03	1.54E+02	-	1.16E+03	4.90E+02	-
18	1.52E+06	1.72E+06	1.17E+02	1.73E+01	-	5.39E+01	5.61E+00	-	**3.31E+01**	**6.10E+00**	-	3.87E+02	1.45E+02	-	3.36E+05	1.46E+05	-	1.74E+06	9.40E+05	+
19	1.97E+04	1.42E+04	7.10E+01	5.98E+00	-	5.07E+01	4.64E+00	-	**2.42E+01**	**6.98E+00**	-	2.03E+02	5.75E+01	-	8.67E+03	3.91E+03	-	1.45E+04	9.40E+03	=
20	1.11E+03	2.68E+02	1.34E+03	2.35E+02	+	1.16E+03	1.22E+02	=	**3.32E+02**	**3.33E+02**	-	9.05E+02	2.79E+02	-	7.80E+02	1.97E+02	-	1.31E+03	2.81E+02	+
21	3.93E+02	4.50E+01	3.17E+02	1.16E+02	-	4.73E+02	1.16E+01	+	4.75E+02	5.98E+01	+	**2.12E+02**	**3.57E+00**	-	4.33E+02	2.14E+01	+	2.93E+02	3.32E+01	-
22	5.61E+03	6.29E+02	**2.14E+03**	**3.56E+03**	-	7.55E+03	5.81E+03	+	2.44E+03	4.09E+03	-	3.48E+03	1.92E+03	-	5.98E+03	9.89E+02	+	6.99E+03	6.55E+03	=
23	6.50E+02	5.01E+01	6.84E+02	1.02E+02	+	6.89E+02	1.78E+01	+	7.16E+02	2.78E+01	+	**4.33E+02**	**1.41E+01**	-	1.07E+03	7.19E+01	+	5.67E+02	3.41E+01	-
24	9.60E+02	9.99E+01	7.42E+02	1.22E+02	-	7.43E+02	1.87E+01	-	8.09E+02	2.32E+01	-	**4.95E+02**	**6.45E+00**	-	1.08E+03	7.07E+01	+	6.78E+02	4.67E+01	-
25	5.56E+02	3.89E+01	4.90E+02	2.67E+01	-	4.81E+02	2.80E+00	-	**4.81E+02**	**2.32E+00**	-	4.85E+02	1.71E+01	-	5.55E+02	2.65E+01	=	6.21E+02	2.61E+01	+
26	3.46E+03	6.03E+02	1.71E+03	1.45E+03	-	3.44E+03	1.71E+02	=	1.19E+03	3.08E+02	-	**8.19E+02**	**3.17E+02**	-	5.45E+03	2.59E+03	+	2.94E+03	5.98E+02	-
27	7.57E+02	9.32E+01	5.22E+02	7.76E+00	-	**5.11E+02**	**1.11E+01**	-	5.32E+02	1.48E+01	-	6.10E+02	5.77E+01	-	1.46E+03	1.70E+02	+	8.69E+02	1.76E+02	+
28	4.97E+02	2.13E+01	4.71E+02	2.15E+01	-	**4.60E+02**	**6.84E+00**	-	**4.60E+02**	**6.84E+00**	-	4.74E+02	2.21E+01	-	4.96E+02	1.79E+01	=	6.27E+02	4.82E+01	+
29	1.23E+03	3.24E+02	1.01E+03	2.25E+02	-	1.13E+03	1.15E+02	-	**6.84E+02**	**3.16E+02**	-	8.44E+02	2.01E+02	-	1.53E+03	2.11E+02	+	1.03E+03	2.48E+02	-
30	1.16E+06	3.76E+05	6.22E+05	3.53E+04	-	**6.01E+05**	**2.98E+04**	-	6.72E+05	6.96E+04	-	6.58E+06	5.27E+06	+	8.75E+05	8.54E+04	-	1.17E+06	3.16E+05	=

Table C.23: Average error ± standard deviation and Wilcoxon signed-rank test (reference: CScDE) for CScDE against EBOwithCMAR, JSO, LSHADE_SPACMA, RB_IPOP_CMA_ES, PPSO, TLBO_FL on CEC-2017 (Awad et al., November 2016) in 100 dimensions.

Function	CScDE		EBOwithCMAR			JSO			LSHADE_SPACMA			RB_IPOP_CMA_ES			PPSO			TLBO_FL		
	Mean	Std	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W
1	1.18E+04	1.29E+04	4.52E+03	2.89E+03	-	3.92E+02	2.47E+02	-	4.31E+03	2.95E+03	-	**2.76E-07**	**5.88E-08**	-	3.58E+06	9.72E+05	+	1.22E+09	9.80E+08	+
2	5.57E+109	3.98E+110	3.08E+31	2.05E+32	-	6.81E+30	4.53E+31	-	9.44E+28	2.72E+29	-	**0.00E+00**	**0.00E+00**	-	1.86E+55	5.98E+55	-	2.05E+110	1.06E+111	+
3	3.01E+05	5.94E+04	1.26E+02	7.17E+01	-	5.20E+01	3.77E+01	-	**3.37E-02**	**2.89E-02**	-	1.42E+01	1.01E+02	-	1.57E+05	1.34E+04	-	2.77E+05	2.45E+04	=
4	2.29E+02	3.67E+01	2.12E+02	1.61E+01	-	2.11E+02	1.38E+01	-	2.20E+02	7.28E+00	=	**1.92E+02**	**3.68E+01**	-	2.84E+02	4.18E+01	+	7.20E+02	1.26E+02	+
5	5.02E+02	6.98E+01	3.39E+02	8.50E+01	-	6.52E+02	2.15E+01	+	**1.22E+01**	**2.99E+00**	-	2.11E+01	4.48E+00	-	5.21E+02	2.30E+01	=	3.50E+02	5.23E+01	-
6	2.98E-02	1.26E-02	1.67E-03	1.72E-04	-	2.78E-04	6.93E-04	-	2.42E-04	5.96E-05	-	**7.95E-05**	**4.41E-04**	-	4.22E+01	2.39E+00	+	1.97E+01	3.01E+00	+
7	6.46E+02	9.03E+01	4.87E+02	2.03E+02	-	7.87E+02	1.97E+01	+	**1.12E+02**	**1.54E+00**	-	1.28E+02	8.99E+00	-	7.46E+02	8.93E+01	+	7.88E+02	9.74E+01	+
8	5.06E+02	7.70E+01	3.67E+02	1.12E+02	-	6.50E+02	1.76E+01	+	**1.02E+01**	**2.98E+00**	-	2.11E+01	4.66E+00	-	5.84E+02	3.07E+01	+	3.78E+02	6.13E+01	-
9	1.79E+04	6.60E+03	1.76E-03	1.25E-02	-	4.59E-02	1.15E-01	-	1.41E-06	4.88E-07	-	**0.00E+00**	**0.00E+00**	-	1.56E+04	9.05E+02	=	2.11E+04	5.18E+03	+
10	1.17E+04	1.21E+03	1.78E+04	1.47E+03	+	2.82E+04	5.79E+02	+	1.66E+04	7.54E+03	=	**8.64E+03**	**2.73E+03**	-	1.15E+04	8.95E+02	=	2.95E+04	5.40E+02	+
11	1.41E+04	1.14E+04	5.52E+02	4.97E+01	-	2.44E+02	7.60E+01	-	**1.69E+02**	**5.82E+01**	-	1.23E+03	2.49E+02	-	1.23E+03	9.36E+01	-	1.08E+03	1.96E+02	-
12	5.10E+06	3.18E+06	3.70E+05	1.12E+05	-	5.55E+04	2.36E+04	-	**1.92E+04**	**7.50E+03**	-	7.20E+04	3.74E+05	-	1.11E+07	2.59E+06	+	3.89E+07	2.49E+07	+
13	8.61E+03	9.28E+03	3.89E+03	1.36E+03	=	6.12E+02	7.05E+01	-	**5.75E+02**	**9.03E+01**	-	6.15E+03	1.34E+03	=	2.38E+03	7.15E+02	-	1.45E+04	6.73E+03	+
14	1.46E+06	8.52E+05	4.66E+02	2.87E+01	-	2.88E+02	1.69E+01	-	**7.33E+01**	**1.24E+01**	-	6.22E+02	8.73E+01	-	4.55E+05	1.48E+05	-	2.53E+06	9.26E+05	+
15	5.99E+03	7.51E+03	6.83E+02	5.15E+01	-	2.24E+02	6.01E+01	-	**2.20E+02**	**5.22E+01**	-	1.44E+03	2.66E+02	-	5.67E+02	1.57E+02	-	3.87E+03	3.56E+03	=
16	4.32E+03	7.17E+02	5.23E+03	8.34E+02	+	6.08E+03	3.31E+02	+	**1.66E+03**	**4.94E+02**	-	1.71E+03	4.65E+02	-	3.21E+03	3.37E+02	-	2.65E+03	5.76E+02	-
17	3.16E+03	5.88E+02	3.91E+03	4.77E+02	+	4.02E+03	2.21E+02	+	3.51E+03	8.61E+02	+	**1.49E+03**	**2.73E+02**	-	2.63E+03	3.34E+02	-	2.25E+03	4.99E+02	-
18	1.76E+06	8.71E+05	1.18E+03	3.19E+02	-	1.80E+02	3.52E+01	-	**1.68E+02**	**3.85E+01**	-	7.08E+02	1.62E+02	-	8.74E+05	2.05E+05	-	5.90E+06	2.00E+06	+
19	5.47E+03	6.56E+03	3.03E+02	3.19E+01	-	1.26E+02	2.18E+01	-	**8.85E+01**	**1.23E+01**	-	8.18E+02	3.31E+02	-	5.47E+02	3.33E+02	-	4.16E+03	5.31E+03	=
20	3.16E+03	6.00E+02	3.89E+03	4.84E+02	+	4.33E+03	2.00E+02	+	**1.54E+03**	**4.68E+02**	-	2.02E+03	3.74E+02	-	2.35E+03	2.58E+02	-	4.55E+03	2.57E+02	+
21	7.42E+02	8.46E+01	5.98E+02	2.68E+02	-	8.78E+02	2.40E+01	+	1.03E+03	1.77E+01	+	**2.46E+02**	**5.20E+00**	-	1.06E+03	5.52E+01	+	6.05E+02	4.82E+01	-
22	1.32E+04	8.63E+02	2.05E+04	5.09E+03	+	2.92E+04	6.49E+02	+	1.00E+04	1.09E+03	-	**9.38E+03**	**3.49E+03**	-	1.38E+04	8.86E+02	+	3.01E+04	4.28E+03	+
23	8.26E+02	6.34E+01	1.00E+03	1.58E+02	+	1.18E+03	1.89E+01	+	1.17E+03	2.05E+01	+	**6.25E+02**	**6.96E+01**	-	2.09E+03	7.84E+01	+	1.17E+03	1.11E+02	+
24	1.37E+03	7.21E+01	1.64E+03	2.39E+01	+	1.48E+03	3.07E+01	+	1.64E+03	2.35E+01	+	**8.93E+02**	**1.41E+01**	-	1.89E+03	1.03E+02	+	2.02E+03	2.61E+02	+
25	7.61E+02	7.34E+01	7.25E+02	3.10E+01	-	7.36E+02	3.52E+01	=	7.10E+02	3.60E+01	-	**6.86E+02**	**4.37E+01**	-	8.02E+02	4.75E+01	+	1.34E+03	1.09E+02	+
26	8.53E+03	7.50E+02	8.94E+03	3.11E+03	+	8.83E+03	5.89E+02	+	3.17E+03	9.83E+01	-	**2.89E+03**	**5.68E+02**	-	1.56E+04	5.18E+03	+	1.09E+04	1.56E+03	+
27	8.64E+02	7.45E+01	5.88E+02	1.53E+01	-	**5.86E+02**	**2.16E+01**	-	5.93E+02	1.54E+01	-	6.73E+02	2.26E+01	-	1.35E+03	9.89E+01	+	1.19E+03	1.49E+02	+
28	5.83E+02	3.22E+01	5.42E+02	2.70E+01	-	5.40E+02	2.41E+01	-	5.29E+02	1.85E+01	-	**5.11E+02**	**6.37E+01**	-	6.36E+02	3.38E+01	+	1.53E+03	2.84E+02	+
29	3.62E+03	5.66E+02	3.56E+03	4.80E+02	=	4.26E+03	2.50E+02	+	**1.94E+03**	**6.82E+02**	-	2.06E+03	4.02E+02	-	3.94E+03	2.98E+02	+	3.49E+03	4.99E+02	=
30	1.18E+04	6.72E+03	3.76E+03	3.55E+02	-	**2.47E+03**	**1.46E+02**	-	2.60E+03	2.24E+02	-	2.87E+04	2.69E+04	+	4.36E+04	1.84E+04	+	8.61E+04	7.69E+04	+

Table C.24: Holm-Bonferroni procedure for CEC-2017 benchmark (Das & Suganthan, 2010) (reference algorithm LSHADE_SPACMA, Rank = 5.84E+00).

j	Optimiser	Rank	z_j	p_j	δ/j	Hypothesis
1	RB_IPOP_CMA_ES	5.16E+00	-2.45E+00	7.14E-03	5.00E-02	Rejected
2	EBOwithCMAR	5.02E+00	-2.96E+00	1.55E-03	2.50E-02	Rejected
3	JSO	4.49E+00	-4.84E+00	6.47E-07	1.67E-02	Rejected
4	PPSO	2.98E+00	-1.02E+01	5.97E-25	1.25E-02	Rejected
5	TLBO_FL	2.53E+00	-1.19E+01	9.25E-33	1.00E-02	Rejected
6	CScDE	2.36E+00	-1.25E+01	4.23E-36	8.33E-03	Rejected

Table D.25: Summary of the BBOB test functions.

Function	**Separable**	**Multimodal**	**Conditioning**	**Function group**
F01	×			Separable functions
F02	×		~1e6	
F03	×	×	~10	
F04	×	×	~10	
F05	×			
F06				Functions with low or moderate conditioning
F07			~100	
F08		$\times^{(*)}$		
F09		$\times^{(*)}$		
F10			~1e6	Functions with high conditioning and unimodal
F11			~1e6	
F12			~1e6	
F13				
F14			~10	
F15		×		Multimodal functions with adequate global structure
F16		×		
F17		×		
F18		×	~1000	
F19		×		
F20		×		Multimodal functions with weak global structure
F21		×	~30	
F22		×	~1000	
F23		×		
F24		×		

Table D.26: Summary of the CEC-2014 test functions.

Function type	**No.**	**Function**	$F_i(x_i^*)$
Unimodal Functions	F01	Rotated High Conditioned Elliptic Function	100
	F02	Rotated Bent Cigar Function	200
	F03	Rotated Discus Function	300
Simple Multimodal Functions	F04	Shifted and Rotated Rosenbrock's Function	400
	F05	Shifted and Rotated Ackley's Function	500
	F06	Shifted and Rotated Weierstrass Function	600
	F07	Shifted and Rotated Griewank's Function	700
	F08	Shifted Rastrigin's Function	800
	F09	Shifted and Rotated Rastrigin's Function	900
	F10	Shifted Schwefel's Function	1000
	F11	Shifted and Rotated Schwefel's Function	1100
	F12	Shifted and Rotated Katsuura Function	1200
	F13	Shifted and Rotated HappyCat Function	1300
	F14	Shifted and Rotated HGBat Function	1400
	F15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
	F16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Functions	F17	Hybrid Function 1 (N = 3)	1700
	F18	Hybrid Function 2 (N = 3)	1800
	F19	Hybrid Function 3 (N = 4)	1900
	F20	Hybrid Function 4 (N = 4)	2000
	F21	Hybrid Function 5 (N = 5)	2100
	F22	Hybrid Function 6 (N = 5)	2200
Composition Functions	F23	Composition Function 1 (N = 5)	2300
	F24	Composition Function 2 (N = 3)	2400
	F25	Composition Function 3 (N = 3)	2500
	F26	Composition Function 4 (N = 5)	2600
	F27	Composition Function 5 (N = 5)	2700
	F28	Composition Function 6 (N = 5)	2800
	F29	Composition Function 7 (N = 6)	2900
	F30	Composition Function 8 (N = 6)	3000
Search Space		$[-100, 100]^D$	

Table D.27: Summary of the chosen CEC-2011 problems.

Problem	**Description**	**Dimensions**	**Constraints**
T_{01}	Parameter Estimation for Frequency-Modulated (FM) Sound Waves	6	Bound constrained
T_{02}	Lennard-Jones Potential Problem	3×10 = 30	Bound constrained
T_{03}	Bifunctional Catalyst Blend Optimal Control Problem	1	Bound constrained
T_{04}	Stirred Tank Reactor Control Problem	1	Unconstrained
T_{05}	Tersoff Potential Function Minimisation Problem	3×10 = 30	Bound constrained