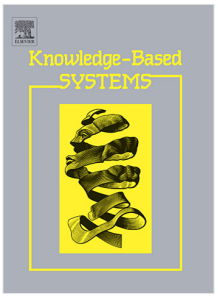

Re-sampled inheritance compact optimization

Giovanni Iacca, Fabio Caraffini

PII: S0950-7051(20)30545-1
DOI: https://doi.org/10.1016/j.knosys.2020.106416
Reference: KNOSYS 106416

To appear in: *Knowledge-Based Systems*

Received date : 2 May 2020
Revised date : 8 July 2020
Accepted date : 19 August 2020

Please cite this article as: G. Iacca and F. Caraffini, Re-sampled inheritance compact optimization, *Knowledge-Based Systems* (2020), doi: https://doi.org/10.1016/j.knosys.2020.106416.

Re-sampled Inheritance Compact Optimization

Giovanni Iacca[a,*], Fabio Caraffini[b]

[a] *Department of Information Engineering and Computer Science, University of Trento, Povo 38123, Italy*
[b] *Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK*

Abstract

Compact optimization is an alternative paradigm in the field of metaheuristics requiring a modest use of memory to optimize a problem. As opposed to population-based algorithms, which conduct the search by employing a set of candidate solutions, compact algorithms use a probabilistic model to describe how solutions are distributed over the search space. Compared to other Estimation of Distribution Algorithms, peculiar features such as the use of simple probabilistic models, in which variables are treated independently, and the need for a minimal number of solutions to be sampled to perform the search, make these algorithms suitable for those applications plagued by memory limitations. Compact algorithms show good results on different kinds of optimization problems but often prematurely converge and perform poorly on non-separable. In this paper, we attempt to overcome these limitations by combining compact algorithms with a restart mechanism named Re-Sampled Inheritance (RI) whose purpose is to avoid premature convergence while also inheriting parts of the variables from the best solution found so far. To assess the effect of the RI mechanism, we extensively test various existing compact algorithms, with and without RI, and compare the best RI-based compact algorithm against several competing algorithms on several optimization problems at different dimensionalities. We also evaluate the effect of the RI parameters on the overall algo-

*Corresponding author
Email addresses: giovanni.iacca@unitn.it (Giovanni Iacca), fabio.caraffini@dmu.ac.uk (Fabio Caraffini)

rithmic performance. Our numerical results not only show that RI consistently enhances the performances of compact algorithms, but also shed some light on the effectiveness of different compact logics at handling problems at different dimensionalities.

Keywords: Compact Optimization, Re-Sampled Inheritance, Differential Evolution, Bacterial Foraging Optimization, Particle Swarm Optimization, Genetic Algorithm.

1. Introduction

One of the most intriguing concepts in computational intelligence is the so-called "compact" optimization [1], that is the study of optimization algorithms that purposely use a limited amount of memory. Compact algorithms are, in practice, an instance of the Estimation of Distribution Algorithms (EDAs) [2], since they use a probabilistic model to describe the distribution of the solutions within the search space, instead of an actual "population" of solutions as done in well-established population-based algorithms such as Evolutionary Algorithms and Swarm Intelligence algorithms. Compared to other EDAs, the distinct features of compact algorithms that make it possible to reduce their memory consumption are 1) the use of a separate distribution for each variable of the problem; 2) the fact that at each iteration they sample a very low number of solutions (typically, three or less), instead of a larger population as done for instance in [3, 4, 5, 6]. To mimic the explorative behaviour of a population of candidate solutions, compact algorithms only require a so-called "Probability Vector" (**PV**). This, can either be an n-dimensional vector if solutions are binary encoded, or a $2 \times n$ matrix if they have a real-valued representation (n is the dimensionality of the problem). More specifically, binary-encoded compact algorithms [7] use as **PV** a single an n-dimensional vector $\mathbf{p} = [p_1, p_2, \ldots, p_n]$ where each $p_i \in [0, 1]$, $i = 1, 2, \ldots, n$, represents the probability that the i-th variable has value 1 (i.e., the relative frequency in a corresponding "virtual population" of N_p solutions, where N_p is a hyper-parameter). In contrast, real-

valued compact algorithms employ a truncated normalized Gaussian Probability Density Function (PDF), described by a **PV** composed of two n-dimensional vectors: a vector of means $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]$ and a vector of variances $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]$, where each i-th element of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ refers to a separate variable.

Compact optimization has been so far adapted to a number of Evolutionary Algorithms, ranging from compact Genetic Algorithm (cGA), either binary [7, 8, 9, 10] or real-valued [11], to compact Differential Evolution (cDE) [12, 13, 14, 15] and compact interactive Memetic Algorithms [16]. Similarly, several Swarm Intelligence algorithms have implemented in a compact fashion, among which compact Particle Swarm Optimization (cPSO) [17], compact Bacterial Optimization (cBFO) [18], compact Teaching-Learning Based Optimization (cTLBO) [19, 20], compact Artificial Bee Colony (cABC) [21, 22], and compact Firefly algorithms [23]. In terms of applications, it is worth noting that while compact algorithms have been designed mainly for (and applied to) memory-limited devices, such as embedded nodes in Wireless Sensor Networks [24] or micro-controllers [10, 12, 25, 26], they have also been successfully applied to cases where memory is not necessarily an issue, such as in industrial plants [17], neural network training [20] and ontology mapping [16].

From an algorithmic point of view, one interesting aspect of compact algorithms is that, compared to population-based algorithms, they are less affected by the typical drawbacks introduced by the presence of "fitter" solutions or distant solutions displaying similar fitness values (as it occurs on a plateau), which have shown to jeopardize the effectiveness of the selection process [27]. Furthermore, the impact of the "virtual" population size is somehow mitigated w.r.t. that of the (actual) population size in population-based algorithms, whose wrong parametrization can lead to premature converge [28] or bias the search [29, 30]. However, the lack of an actual population also comes with drawbacks: since diversity cannot be maintained, exploration, which is key to solve highly multimodal problems, is limited. Furthermore, since compact algorithms process each variable separately, they tend to fail at solving fully non-separable problems. For this class of problems, population-based algorithms might still

better-suited [31].

To overcome these limitations, here we investigate the use of a restart mechanism named Re-Sampled Inheritance (RI) [32, 33, 34] in compact optimization. The main idea of RI is to recombine a randomly generated solution with the best solution, found so far by the compact algorithm, through the crossover used in Differential Evolution [35, 36]. Of note, this mechanism is similar to the re-sampling used in Iterated Local Search (ILS) [37] and other multi-stage algorithms [38]. Indeed, a compact algorithm with RI can be seen as an instance of ILS, as it shares the idea of performing restart by applying a small perturbation to the best solution: a perturbation as small as possible to not disrupt the search too much, but as large as needed to allow the search to converge at every restart to a different optimum. However, it should be noted that ILS has been mostly applied to combinatorial optimization, while here we focus explicitly on continuous optimization.

Previous works [32, 34, 39] have demonstrated that RI is a simple yet effective way to improve the performance of an optimization algorithm, for instance when applied to restart CMA-ES [40] or CMA-ES 1+1 [41]. Here, in the light of the Ockham's razor [38] we regard "simplicity" as a positive attribute of an optimization algorithm, since simpler algorithms are e.g. easier to understand, tune (if needed), port to different (possibly memory-limited) hardware and/or different programming languages, etc. In this sense, the degree of "simplicity" of an algorithm can be measured at different levels, such as: a) simplicity in terms of numerical complexity (that can be approximated e.g. by the overhead added by the algorithm to the time needed for randomly sampling a given number of candidate solutions); b) simplicity in terms of memory needed to store all data structures used by the algorithm (for which one possible proxy is e.g. the number of n-dimensional vectors stored by the algorithm, where n is the problem dimension); c) simplicity in terms of implementation (for which one could use e.g., as a proxy, the number of lines of code and/or the size of the binary file implementing the algorithm). However, while the first two measures are objective characteristics of an algorithm, the last aspect heavily depends on

implementation details such as programming language, use of external classes or methods, etc. Therefore, we consider this last measure somehow less reliable.

Apart from its simple algorithmic design, the main advantage of RI is that it inherently allows escaping from local optima while preserving pieces of information from the current best (**elite**). The intuition is then that these same features have the potential to allow compact algorithms to escape from local optima, where they might get stuck because of lack of diversity, but also to obtain better results on non-separable problems, by inheriting groups of variables from the **elite**. In addition to that, it is important to highlight that the RI mechanism does not require any additional memory w.r.t. to an un-restarted compact algorithm, thus maintaining a low memory consumption.

In our previous study [42], we performed a preliminary analysis of the effect of RI on compact optimization, finding that 1) RI tends to improve the performance of compact algorithms; 2) compared to a uniform random restart (without inheritance), RI produces better results; 3) compared to pure random search/random walk, compact algorithms with RI perform better. Still, several research questions remained unanswered, which we try to address in this paper. In particular, here we aim to 1) evaluating how the performance of different compact algorithms, with and without RI, scales with the problem dimensionality; 2) evaluating if some compact logics are better suited than other for different dimensionalities; 3) comparing the RI-based compact algorithms against some of the state-of-the-art algorithms for continuous optimization to assess if, by using RI, the compact algorithms can obtain good performances also when compared to memory-consuming population-based algorithms; 4) evaluating the effect of the parametrization of the RI mechanism.

In order to reach these goals, we apply RI separately to four existing compact algorithms for continuous optimization, namely cDE "light" [14], cGA [11], cPSO [17], and cBFO [18], and perform extensive tests on the CEC2014 benchmark [43] on various dimensionalities, up to 100 dimensions, and the CEC2013-LSGO benchmark [44] in 1000 dimensions. For each problem in the two benchmarks, we also compare the compact algorithms with and without RI with

various algorithms from the state-of-the-art in continuous optimization, both single-solution and population-based. The numerical details of these experiments are reported in Appendix B-Appendix D. In additional experiments (reported in 4.1), we also test the compact algorithms with and without RI on three selected problems from the CEC2011 benchmark on real-world optimization problems [45], with different dimensionalities. Finally, we report in Appendix E an analysis of the effect of the RI parametrization on the algorithmic performance. Overall, our broad experimental setup is conceived to analyze the performance of the compact algorithms with and without RI in relation to the problem dimensionality, fitness landscape properties (in particular, separability and modality), and lastly the RI parametrization. To summarize, our main contributions can be identified in these three aspects:

- The first main contribution of this paper is on the algorithmic side: we show how the RI mechanism can be added to several different compact algorithms, both belonging to Evolutionary Computation and Swarm Intelligence, and also answer the question as to whether some compact logics are more effective than others on some specific problems and at some specific dimensionalities.

- The second main contribution is experimental: we have conducted an extensive experimental campaign that includes 30 functions from the CEC2014 in 10, 50, and 100D, 15 functions in 1000D from CEC2013-LSGO benchmark, and three industrial applications with three different dimensionalities.

- The third contribution is on the parameter analysis: we performed a thorough analysis on the effect of the parameters of the RI mechanism as well as the specific crossover operator used in it on the overall performance of the algorithms.

These three contributions are unprecedented in the literature and not only they provide several novel insights for practical uses of compact algorithms with RI, but also they open up several interesting future research directions as we highlight in the conclusions.

The rest of this paper is organized as follows: Section 2 presents the background concepts on compact optimization. Section 3 describes the general algorithmic framework which combines compact algorithms with Re-Sampled Inheritance. Our experimental setup and the numerical results are then presented in Section 4. Finally, Section 5 concludes this work.

2. Background on Real-valued Compact Optimization

In the remainder of this paper, we will focus on real-valued compact algorithms, i.e. algorithms for solving continuous optimization where each variable is encoded directly as a real-value rather than with a binary encoding as in the original binary cGA [7]. As discussed earlier, real-valued compact algorithms model each i-th variable with a Gaussian PDF, truncated in the interval $[-1, 1]$, and normalized such that its area is unitary (see Figure 1 for examples of truncated normalized PDFs). Therefore, $\mathbf{PV}$ is set as $[\boldsymbol{\mu}, \boldsymbol{\sigma}]$. It must be remarked that a truncated PDF is used because, by removing the infinite tails, the sampling always occurs in a finite interval ($[-1, 1]$) that can be easily mapped to the search domain. Furthermore, a truncated PDF can be easily re-scaled to guarantee a unitary area, such that the overall sampling probability sums up to one.

At the beginning of the compact optimization process, each i-th variable is associated with $\mu_i = 0$ and $\sigma_i = \lambda$, where λ is a large positive constant (e.g. $\lambda = 10$). This initial setting makes sure that the algorithm is capable, at the early stages of the optimization process, of exploring the search space the same way a uniformly sampled population would do. Indeed, with a high σ_i value the truncated normalized Gaussian PDF is a good approximation of a uniform distribution $\mathcal{U}(-1, 1)$. This is graphically shown in Figure 1. Unless differently specified, the first solution is sampled with this initial setting and stored in memory as the $\mathbf{elite} = (elite_1, elite_2, \ldots, elite_n)$.

The sampling mechanism is a common denominator for all compact algorithms as it follows a consolidated logic. First, for each i-th variable the corre-

sponding truncated normalized Gaussian PDF is calculated as follows:

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

where erf() is the error function [46], and μ_i and σ_i are taken from $\mathbf{PV}$. The PDF is subsequently processed into a Cumulative Density Function (CDF), by means of the Chebyshev polynomials approximation process described in [47]. Then, a random number r is sampled from a uniform distribution $\mathcal{U}(0,1)$ and finally the i-th variable x_i is obtained by evaluating the inverse CDF corresponding to r, as graphically shown in Figure 2. This sampling process is performed n times, thus producing n variables x_i $(i = 1, 2, \ldots, n)$ to form a complete solution.

After the first sampling of **elite**, the iterative process starts. At each iteration, depending on the specific compact algorithm, a candidate solution $\mathbf{x}$ is generated by sampling one or more solutions from the current $\mathbf{PV}$, according to the sampling mechanism just described, and applying the algorithm-specific operators such as mutation and crossover. The number of sampled solutions depends on the specific case: for instance, three solutions are required if the "rand/1" mutation typical of Differential Evolution has to be reproduced in the cDE algorithm (namely $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, such that a new solution $\mathbf{x}$ can be found by: $\mathbf{x} = \mathbf{x}_1 + F(\mathbf{x}_2 - \mathbf{x}_3)$, where F is the scale factor parameter used in DE). In the case of cGA, only one solution must be sampled to be recombined with the current **elite** through crossover. Details on the compact algorithms involved in this study can be found in the original papers ([11, 14, 17, 18], respectively for cGA, cDE, cPSO, and cBFO). For reference, we report in Appendix C the pseudo-code of the algorithms we used in our experimentation.

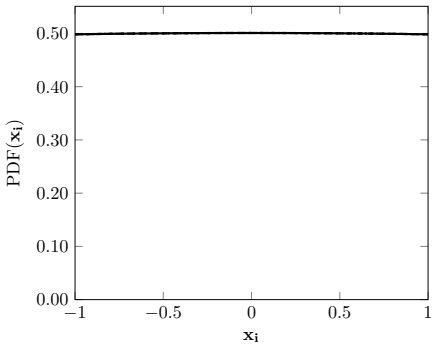

Figure 1: Truncated Gaussian PDF for $\mu = 0.3$ and $\sigma = 0.4$ (smooth line), $\mu = 0$ and $\sigma = 1$ (dashed line) and $\mu = 0$ and $\sigma = 10$ (dotted line).

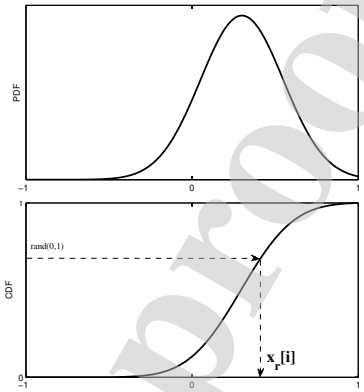

Figure 2: Sampling mechanism: PDF and corresponding CDF.

Regardless of the specific compact algorithm, at the end of each iteration, the newly generated solution is evaluated and its fitness is compared against that of the **elite** solution: the solution displaying the best fitness value among the two, referred to as **winner**, and the one displaying the worst, referred to as **loser**, are then used to update **PV** as follows:

$$\mu_i^{new} = \mu_i^{old} + \frac{1}{N_p}\left(winner_i - looser_i\right) \tag{2}$$

$$\sigma_i^{(new)} = \sqrt{\left(\sigma_i^{old}\right)^2 + \left(\mu_i^{old}\right)^2 - \left(\mu_i^{new}\right)^2 + \frac{1}{N_p}\left(winner_i^2 - looser_i^2\right)} \tag{3}$$

where it can be observed that for each i-th variable the distance between the corresponding elements of **winner** and **loser** is used to change μ_i and σ_i so to "shift" the PDF towards the most promising basin of attraction and "shrink" it around it, such that the next sampled solutions will be biased towards that area of the search space (see Figure 1). The virtual population size, N_p, regulates the strength of the update (the larger N_p, the smaller the updates).

Finally, the last step of each iteration is to update the **elite**. This update depends on the chosen elitism scheme: persistent, if the replacement occurs only when the current elite is outperformed; non-persistent, if **elite** is replaced

in case of improvement, but also when no improvements are found for η consecutive iterations of the algorithm (in this case **elite** is reinitialized with a newly generated solution). The general structure of a compact optimization algorithm is shown in Algorithm 1.

Algorithm 1 General structure of a compact algorithm

```
1: initialize PV                          ▷ μ_i = 0, σ_i = λ for each i-th variable
2: generate elite by means of PV
3: while stop condition is not met do
4:     generate candidate solution x      ▷ depending on the compact algorithm
5:     compare fitness of x and elite
6:     update PV                          ▷ see Equations (2) and (3)
7:     if x outperforms elite then        ▷ depending on the elitism scheme
8:         elite = x
9:     end if
10: end while
11: return elite
```

3. Compact Optimization Algorithms with Re-Sampled Inheritance

Algorithm 2 shows the proposed optimization framework in its most general formulation. Several different implementations of this scheme can be obtained by simply using a specific perturbation logic in the internally used compact optimizer. This only requires writing a modest portion of code to process the sampled candidate solutions into $\mathbf{x}$ – see line 4 of Algorithm 1 and the pseudo-codes in the Appendix. It is worth noting that every time a compact algorithm is executed inside Algorithm 2 the **elite** is not randomly initialized, as in line 2 in Algorithm 1, but passed externally. The RI mechanism is activated at the end of each execution of the compact algorithm (that is allotted a given % of the total budget), and proceeds as follows. First, it randomly generates a solution $\mathbf{x}^{rand}$ from a uniform distribution within the given search space.

Then, it recombines $\mathbf{x}$ with the current **elite** obtained so far by the algorithm, by applying the exponential crossover used in Differential Evolution, see the pseudo-code in Algorithm 3. This starts by randomly selecting an initial index i between 1 and n and copying the corresponding variable from **elite** into $\mathbf{x}$. Then, starting from the initial index, the subsequent variables from **elite** are copied into the corresponding positions of $\mathbf{x}$ as long as a random number r sampled from a uniform distribution $\mathcal{U}(0,1)$ is less than or equal to the crossover rate Cr. As soon as $r > Cr$, the copy stops. The Cr parameter affects the expected number of variables inherited from **elite**, in particular is set as $Cr = 1/\sqrt[n\alpha]{2}$, where $n\alpha$ is the expected number of variables that are copied from **elite** [14]. Of note, during the copy the solutions are handled as cyclic buffers, i.e. when the n-th variable is reached the next to be copied is the 1st variable. At the end of the copy process, the fitness of $\mathbf{x}$ is compared with that of **elite**. If the newly generated solution outperforms **elite**, an **elite** replacement occurs and the compact algorithm is restarted from this updated **elite**. If no improvement is found, the **elite** is not updated. After each restart, regardless of the fact that RI led or not to an improvement of **elite**, the compact algorithm is reinitialized with $\mu_i = 0$ and $\sigma_i = \lambda$ for each i-th variable, see line 1 in Algorithm 1.

As we discussed earlier, the idea behind the RI mechanism is to restart the compact algorithm from a "partially random" solution, i.e. a solution that is randomly generated but still inherits part of the variables from the current **elite**. This way the restart is not entirely disruptive, as it preserves, on average, $n\alpha$ variables. This form of partial inheritance allows the algorithm to keep some information after each restart, but also to escape from local optima.

4. Experimental results

In the following we present the numerical results obtained on the CEC2014 [43] and CEC2013-LSGO [44] benchmarks[1]. Both benchmarks are composed

[1]All the experiments reported here and in the Appendices have been performed on a workstation with 64 CPUs (AMD Opteron™ 6378 @2.4GHz) using the Stochastic Optimisation

Algorithm 2 Compact algorithm with Re-Sampled Inheritance

generate **elite** from a uniform distribution
while stop condition is not met **do**
    ▷ Compact algorithm
    run compact algorithm for a % of the total budget ▷ Algorithm 1
    ▷ Re-Sampled Inheritance
    generate $\mathbf{x}^{rand}$ from a uniform distribution
    generate $\mathbf{x}$ by crossover of $\mathbf{x}^{rand}$ and **elite** ▷ Algorithm 3
    if $\mathbf{x}$ outperforms **elite then**
        elite $= \mathbf{x}$
    end if
end while
return elite

Algorithm 3 Exponential crossover

Input: two parents $\mathbf{x}^A$ and $\mathbf{x}^B$ ▷ each one containing n variables
$\mathbf{x}^{offspring} = \mathbf{x}^A$
generate a random index i between 1 and n
$x_i^{offspring} = x_i^B$ ▷ exchange the i-th variable
$k = 1$
while $r \sim \mathcal{U}(0,1) \leq Cr$ **and** $k < n$ **do** ▷ exchange the following variables
    $i = i + 1$
    if $i == n$ **then**
        $i = 1$
    end if
    $x_i^{offspring} = x_i^B$ ▷ exchange the i-th variable
    $k = k + 1$
end while
return $\mathbf{x}^{offspring}$

Software (SOS) platform [48] available at: `https://github.com/facaraff/SOS`.

of functions with different properties in terms of separability, ill-conditioning, and landscape structure. In particular, the CEC2014 benchmark contains 30 functions, which are all non-separable except for f_8 (Shifted Rastrigin Function) and f_{10} (Shifted Schwefel Function). Therefore this benchmark is particularly suited for testing the performance of optimization algorithms on non-separable problems. In contrast, the CEC2013-LSGO benchmark presents functions with variable levels of separability. In fact, it contains 15 functions, of which four are fully separable (f_1: Elliptic Function, f_2: Rastrigin Function, f_3: Ackley Function, and f_{12}: Overlapping Rosenbrock's Function), one is fully non-separable (f_{15}: Schwefel Problem 1.2), whereas all the other functions are defined as partially separable. Therefore, this benchmark is suited for testing how an algorithm's performance, in large dimensionalities, changes with the level of problem's separability.

To solve both benchmarks, we considered the following four real-valued compact algorithms, with the parametrization proposed in their original papers:

- cDE "light" [14] (in the following simply referred to as cDE) with exponential crossover and parameters: $N_p = 300$, $F = 0.5$, and $\alpha_m = 0.25$;
- cBFO [18], with parameters: $N_p = 300$, $C = 0.1$, $\tau = 0.1$ and $N_s = 4$;
- cGA [11], with persistent elitism and parameters: $N_p = 300$;
- cPSO [17], with parameters: $N_p = 300$, $\phi_1 = 0.2$, $\phi_2 = 0.07$, $\phi_3 = 3.74$, $\gamma_1 = 1$, and $\gamma_2 = 1$.

As for the equivalent versions with RI, for each of the four aforementioned compact algorithms, we used the same parametrization as in the versions without RI. In all cases, the RI component was parametrized with $\alpha = 0.05$ (such that only 5% of the variables are inherited, on average, from **elite**), with each run of the compact algorithm between restarts executed for a *local budget* of 25% of the total budget (an analysis of these two parameters is reported in Appendix E).

To assess the scalability of all the algorithms, we performed experiments in 10, 50 and 100 dimensions on the CEC2014 benchmark and 1000 dimensions on the CEC2013-LSGO benchmark. Thus, the total experimental setup consists of eight compact algorithms and $30 \times 3 + 15 = 105$ optimization problems (i.e. 30 CEC2014 functions each tested in three different dimensionalities, plus 15 CEC2013-LSGO functions). On each benchmark function, each algorithm was executed for 30 independent runs, to collect statistics on the fitness values obtained in each run at the end of the allotted computational budget. Each run was executed for a total budget of $5000 \times D$ function evaluations, being D the problem dimension.

For the sake of brevity, we report here only the results of the sequentially rejective Holm-Bonferroni procedure [49, 50] applied to the eight algorithms on all problems from the CEC2014 and CEC2013-LSGO benchmarks. The detailed results in terms of average error $\pm$ standard deviation and pairwise Wilcoxon Rank-Sum test [51] are reported in Appendix B.

Before discussing the results, let us describe how the Holm-Bonferroni procedure works. This procedure consists of the following: considering the average fitness values obtained by all the algorithms at the end of the computational budget, for each problem a score R_i is assigned to each algorithm, for $i = 1, 2, \ldots, N_A$ (where N_A is the number of algorithms under analysis), being N_A the score of the algorithm displaying the best performance on that problem, $N_A - 1$ the score of the second-best, and so on. The algorithm displaying the worst performance scores 1. These scores are then averaged, for each algorithm, over the whole set of test problems. The algorithms are sorted based on these average scores. Indicating with R_0 the Rank of the algorithm displaying the highest average score, which is considered as the *reference algorithm*, and with R_j for $j = 1, 2, \ldots, N_A - 1$ the Ranks of the remaining $N_A - 1$ algorithms, the values z_j are calculated as:

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}} \tag{4}$$

where N_{TP} is the number of test problems in consideration. By means of the

z_j values, the corresponding cumulative normal distribution values p_j are derived. These p_j values are then compared to the corresponding δ/j where δ is the confidence interval, set to 0.05: if $p_j < \delta/j$, the null-hypothesis (that the reference algorithm has the same performance as the j-th algorithm) is rejected, otherwise is accepted as well as all the subsequent tests.

Table 1 displays the Ranks, z_j values, p_j values, and corresponding δ/j obtained by this procedure where RIcDE was set as the reference algorithm (since it has the highest Rank, 6.54 out of 8), with $N_A = 8$ and $N_{TP} = 105$. Moreover, the table indicates whether the null-hypothesis is rejected or not. In this case, all the hypotheses are sequentially rejected, meaning that RIcDE has a statistically better overall performance (i.e., on the entire set of tested problems) than all the other algorithms under study. Moreover, the Ranks show that each compact algorithm with RI performs better (on average) than the corresponding algorithm without RI, indicating the fact that the RI component is beneficial to all the compact algorithms considered in our experimentation.

Table 1: Holm-Bonferroni procedure (reference: RIcDE-light, Rank=6.54e+00) on CEC2014 in 10, 50 and 100D and CEC2013-LSGO in 1000D.

j	Algorithm	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcBFO	5.24e+00	-4.38e+00	6.04e-06	5.00e-02	Rejected
2	RIcGA	5.23e+00	-4.41e+00	5.21e-06	2.50e-02	Rejected
3	cDE	4.84e+00	-5.72e+00	5.39e-09	1.67e-02	Rejected
4	cBFO	4.71e+00	-6.13e+00	4.31e-10	1.25e-02	Rejected
5	cGA	3.65e+00	-9.71e+00	1.35e-22	1.00e-02	Rejected
6	RIcPSO	2.80e+00	-1.26e+01	1.89e-36	8.33e-03	Rejected
7	cPSO	2.78e+00	-1.26e+01	8.42e-37	7.14e-03	Rejected

Further considerations can be drawn by analysing Tables B.11-B.18 (in Appendix B), which report detailed numerical results on each considered problem in terms of: 1) average error w.r.t. the known optimal fitness ± standard deviation; 2) pairwise comparisons -according to the non-parametric Wilcoxon

Rank-Sum test- between RIcDE, taken as reference for being the best algorithm according to the Holm-Bonferroni procedure shown in Table 1, and the other seven algorithms considered in our study. More specifically, Tables B.11, B.12 and B.13 report the results on the CEC2014 benchmark in 10, 50 and 100 dimensions, respectively, while Table B.14 reports the results on the CEC2013-LSGO benchmark in 1000 dimensions, for the four compact algorithms (without RI), compared against RIcDE. Instead, Tables B.15, B.16 and B.17 report the results on the CEC2014 benchmark in 10, 50 and 100 dimensions, respectively, while Table B.18 reports the results on the CEC2013-LSGO benchmark in 1000 dimensions, for the three other compact algorithms (with RI), here dubbed as RIcBFO, RIcGA and RIcPSO, again compared against RIcDE. In each of these tables, the boldface indicates the algorithm that obtains the minimum average error on each tested benchmark function. The result of each pairwise Wilcoxon Rank-Sum test is indicated as '+', '-', or '=', representing, respectively, the fact that the reference algorithm (RIcDE) is better than/worse than/equal to the algorithm corresponding to the focal column label. In all the pairwise tests, we considered a confidence interval of 0.05.

From the comparison with the four compact algorithms (without RI) we can draw a first important conclusion: RIcDE consistently outperforms (except f_1 in 10 dimensions) cDE, cGA and cPSO on all the CEC2014 benchmark functions in 10, 50 and 100 dimensions, see Tables B.11-B.13. Among the compact algorithms (without RI), cBFO results more competitive than the others when compared with RIcDE, especially in large scale problems (50 and 100 dimensions), and particularly on the second part of the CEC2014 benchmark (hybrid and composition functions). On the other hand, RIcDE obtains better solutions than cBFO on most of the functions in the first part of the CEC2014 benchmark (unimodal and multimodal functions). As for the CEC2013-LSGO benchmark in 1000 dimensions, see Table B.14, an interesting fact emerges: in this case, cPSO, which was consistently outperformed by the other algorithms in lower dimensionalities, performs much better than both the other three compact algorithms (without RI) and RIcDE, obtaining the lowest error in 10 out of 15

functions, and resulting statistically superior to RIcDE in 10 cases out of 15. We will come back to this aspect shortly.

From the comparison with RIcBFO, RIcGA and RIcPSO (see Tables B.15-B.18), we can note the following. As for the comparison with RIcBFO and RIcGA, we can observe again that the cBFO-based algorithm is more competitive especially on medium-high dimensionalities (50 and 100) on the hybrid and composition functions of the CEC2014 benchmark. On the other hand, RIcGA seems to be somehow competitive only in 10 dimensions, while its performance degrades progressively as the dimensionality increases (with a few exceptions at 1000 dimensions). More interesting is the comparison with RIcPSO: similarly to the comparison against cPSO without RI, also in this case RIcDE consistently outperforms RIcPSO for lower dimensionalities (10, 50 and 100), while for 1000 dimensions RIcPSO performs on average much better than both RIcDE and the other two compact algorithms with RI, obtaining the lowest error in 9 out of 15 functions and resulting statistically superior to RIcDE in 10 cases out of 15.

The trend regarding PSO -either with or without RI- seems then confirmed: it performs quite poorly up to 100 dimensions, while for very large scale problems (1000) its performance emerges. This is also confirmed by the Holm-Bonferroni procedure in Table 1, where cPSO turns out to be the least efficient of all the compact algorithms, regardless the use of RI (although it can be seen that RIcPSO outperforms the original cPSO). We collected further evidence on this aspect by performing two additional Holm-Bonferroni procedures, this time separating the analysis for the CEC2014 problems up to 100 dimensions from that for the CEC2013-LSGO problems at 1000 dimensions, see Tables 2 and 3 respectively. In particular, in Table 2 RIcDE is set as reference algorithm (having the highest Rank, 6.92 out of 8), with $N_A = 8$ and $N_{TP} = 90$. This table is practically equivalent to Table 1, with the same ranking and decisions on the null-hypothesis: this mostly depends on the fact that it is based on 90 out of 105 problems, i.e. almost the totality of the problems included in Table 1. On the other hand, in Table 3 we only consider the 15 CEC2013-LSGO problems at 1000 dimensions. Of note, the reference algorithm is cPSO (having

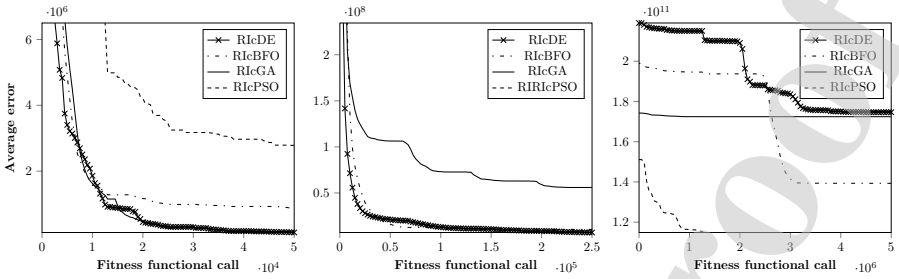

Figure 3: Average error trend on f_1 from CEC2014 in 10 and 50D, and f_1 from CEC2013-LSGO in 1000D (from left to right).

the highest Rank, 6.27 out of 8), with $N_A = 8$ and $N_{TP} = 15$. We can see immediately that the results at very large scales are drastically different from the lower dimensionalities. Indeed, cPSO, RIcPSO and RIcGA (in this order) are the best-ranked algorithms, with accepted null-hypothesis on their statistical equivalence. All the other algorithms - including RIcDE which was top-ranked on CEC2014 up to 100 dimensions - are evaluated as statistically inferior to the first three algorithms.

Table 2: Holm-Bonferroni procedure (reference: RIcDE, Rank=6.92e+00) on CEC2014 in 10, 50 and 100D.

j	Algorithm	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcBFO	5.39e+00	-4.76e+00	9.61e-07	5.00e-02	Rejected
2	RIcGA	5.30e+00	-5.04e+00	2.36e-07	2.50e-02	Rejected
3	cDE	5.10e+00	-5.66e+00	7.63e-09	1.67e-02	Rejected
4	cBFO	5.04e+00	-5.83e+00	2.75e-09	1.25e-02	Rejected
5	cGA	3.52e+00	-1.06e+01	2.33e-26	1.00e-02	Rejected
6	RIcPSO	2.28e+00	-1.44e+01	1.87e-47	8.33e-03	Rejected
7	cPSO	2.20e+00	-1.47e+01	5.49e-49	7.14e-03	Rejected

To further highlight this finding, we show in Figure 3 the average error trend across 30 independent runs of each compact algorithm with RI (RIcDE, RIcBFO, RIcGA and RIcPSO) on function f_1 (Rotated High Conditioned El-

Table 3: Holm-Bonferroni procedure (reference: cPSO, Rank=6.27e+00) on CEC2013-LSGO in 1000D.

j	Algorithm	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcPSO	5.93e+00	-4.23e-01	3.36e-01	5.00e-02	Accepted
2	RIcGA	4.80e+00	-1.86e+00	3.15e-02	2.50e-02	Accepted
3	cGA	4.40e+00	-2.37e+00	8.98e-03	1.67e-02	Rejected
4	RIcBFO	4.33e+00	-2.45e+00	7.12e-03	1.25e-02	Rejected
5	RIcDE	4.27e+00	-2.54e+00	5.61e-03	1.00e-02	Rejected
6	cDE	3.27e+00	-3.80e+00	7.14e-05	8.33e-03	Rejected
7	cBFO	2.73e+00	-4.48e+00	3.74e-06	7.14e-03	Rejected

liptic Function) from the CEC2014 benchmark in 10 and 50 dimensions, and f_1 (Shifted Elliptic Function) from the CEC2013-LSGO benchmark in 1000 dimensions[2]. From the figure it can be seen that RIcDE outperforms the other three algorithms in 10 and 50 dimensions (with RIcPSO performing especially poorly in the 10D case), while when the dimensionality grows to 100D the situation changes, with RIcPSO rapidly converging to a much lower error w.r.t. the three other algorithms. This observation is a further indication of how different compact logics can behave quite differently depending on the problem dimensionality, with the cPSO logics (with or without RI) that appears more suitable at higher dimensionalities (larger than 100D), whereas for lower dimensionalities cDE (with RI) appears a better choice.

In order to assess how the best compact algorithm with RI (RIcDE) performs

[2]It should be noted that while this function is formulated in the same way in the two benchmarks, in the CEC2014 it is also rotated, with an additional bias. This results in f_1 being a unimodal function with quadratic ill-conditioning in both benchmarks, separable according to the CEC2013-LSGO definition, and non-separable (because of the rotation) according to the CEC2014 definition. The different shifts used in the two benchmarks and the presence of a bias factor in the CEC2014 definition also makes the function slightly different. Despite these differences, the behavior of the compact algorithms on this problem is fundamentally the same and what we would like to highlight here is the effect of the dimensionality.

Table 4: Comparison against single-solution algorithms: Holm-Bonferroni procedure (reference: RIcDE, Rank=3.03e+00) on CEC 2014 in 10, 50 and 100D.

j	Optimizer	Rank	z_j	p_j	α/j	Hypothesis
1	nuSA	2.49e+00	-3.65e+00	1.30e-04	5.00e-02	Rejected
2	CMA-ES (1+1)	2.30e+00	-4.92e+00	4.34e-07	2.50e-02	Rejected
3	PMS	2.18e+00	-5.74e+00	4.76e-09	1.67e-02	Rejected

against some algorithms from the state-of-the-art, we also compared its results on the CEC2014 benchmark (for 10, 50 and 100 dimensions) against a set of state-of-the-art algorithms from the literature, which we divide into two categories: single-solution and population-based. For each category we identified three representatives of different algorithmic logics.

As for the single-solution algorithms, we considered the following competitors:

- Non-Uniform Simulated Annealing (nuSA) [52]
- Covariance Matrix Adaptation Evolution Strategy 1+1, (CMA-ES 1+1) [41]
- Parallel Memetic Structures (PMS) [53]

These algorithms have been chosen as representatives of three different algorithmic logics, namely Simulated Annealing, Evolution Strategies, and Memetic Computing. For the comparisons, we parametrized them as suggested in their original papers. As for RIcDE, it was parametrized as in the previous experiments (exponential crossover, $N_p = 300$, $F = 0.5$, $\alpha_m = 0.25$, $\alpha = 0.05$, local budget=25%). In Table 4, we report the results of the Holm-Bonferroni procedure applied to the four aforementioned algorithms on the CEC2014 benchmark in three dimensionalities, see Appendix C for detailed results in terms of average error $\pm$ standard deviation and pairwise Wilcoxon Rank-Sum test.

The numerical results indicate that RIcDE is highly competitive against the other single-solution algorithms, including the well-established CMA-ES 1+1

[41]. While there is no clear trend on the pairwise Wilcoxon test (in fact, it seems that each algorithm tends to perform better than the others on some isolated cases), the Holm-Bonferroni procedure in Table 4 indicates that the global performance of RIcDE is superior to that of the three other single-solution algorithms, since the null-hypothesis that RIcDE is statistically equivalent to the other algorithms is rejected in all the three cases.

Concerning the population-based algorithms, we considered the following competitors:

- Adaptive Differential Evolution With Optional Archive (JADE) [54]
- Comprehensive Learning Particle Swarm Optimization (CLPSO) [55]
- Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search (EPSDE-LS) [56]

Also in this case, we have chosen these algorithms as representatives of three different algorithmic logics, namely Differential Evolution, Particle Swarm Optimization, and Memetic Computing [57]. For the comparisons, we parametrized them as suggested in their original papers, with RIcDE parametrized as in the previous experiments (exponential crossover, $N_p = 300$, $F = 0.5$, $\alpha_m = 0.25$, $\alpha = 0.05$, local budget=25%). In Table 5, we report the results of the Holm-Bonferroni procedure applied to the four algorithms on the CEC2014 benchmark in three dimensionalities, see Appendix D for detailed results in terms of average error $\pm$ standard deviation and pairwise Wilcoxon Rank-Sum test.

Overall, from the Wilcoxon Rank-Sum test and the Holm-Bonferroni procedure it results that, among the tested algorithms, EPSDE-LS performs best at all the three dimensionalities. On the other hand, it is interesting to note from the Holm-Bonferroni procedure that the null-hypothesis on statistical equivalence on the whole set of problems is accepted on the comparison between RIcDE, EPSDE-LS and CLPSO, especially considering that the latter two are much more complex algorithms. Even more surprising is that a state-of-the-art, memory-hungry population-based algorithm like JADE (which, on top of the

Table 5: Comparison against population-based algorithms: Holm-Bonferroni procedure (reference: RIcDE, Rank=2.60e+00) on CEC 2014 in 10, 50 and 100D.

j	Optimizer	Rank	z_j	p_j	α/j	Hypothesis
1	EPSDE-LS	3.64e+00	7.01e+00	1.00e+00	5.00e-02	Accepted
2	CLPSO	2.68e+00	5.22e-01	6.99e-01	2.50e-02	Accepted
3	JADE	1.08e+00	-1.02e+01	8.82e-25	1.67e-02	Rejected

population, also uses an archive of solutions) is instead statistically inferior to a low-memory algorithm like RIcDE. However, on the problems where RIcDE is outperformed, the error on the fitness can be quite larger than that obtained by population-based algorithms (few orders of magnitude), which indicates that, if high accuracy is needed during the optimization and memory is available, the best population-based algorithms should be preferred.

4.1. Results on three selected problems from CEC2011

In another set of experiments, we also validated the applicability of RIcDE on three selected real-world problems taken from the CEC2011 benchmark on real-world applications [45], namely P_1, P_2 and P_7, respectively with 6, 30 and 20 parameters. A brief description of the three problems follows:

- P_1 (Parameter Estimation for Frequency-Modulated (FM) Sound Waves): This problem consists in optimizing 6 parameters which describe a wave signal that correspond to generating a sound similar to a given target sound. The fitness function is the sum of the squared errors between the estimated wave and the target wave. This problem is highly multimodal and has a strong epistasis.

- P_2 (Lennard-Jones Potential Problem): This problem consists in finding the position of N atoms to form the Lennard-Jones cluster [58] having the lowest molecular potential energy. The resulting optimization problem requires to find the three coordinates x, y and z for each atom ($N = 10$ in this study,

thus resulting into a 30-dimensional problem) and displays a number of local minima increasing exponentially with N.

- P_7 (Spread Spectrum Radar Polyphase Code Design): The objective of this problem is to minimize the module of the biggest sample of an auto-correlation function related to the complex envelope of a compressed radar pulse, with the 20 design variables representing symmetrical phase differences. This problem belongs to the class of the continuous min–max non-linear non-convex global optimization problems and is characterized by a piece-wise smooth objective function with numerous local optima.

Of note, these three problems are all non-separable and multimodal. On these problems, we compared the compact algorithms analyzed in this study, with and without RI, parametrized as discussed earlier.

The numerical results in terms of average fitness ± standard deviation and Wilcoxon Rank-Sum test over 30 independent runs of each algorithm on each problem are shown in Tables 6-7, respectively for RI-cDE against four compact algorithms without RI, and RI-cDE against three other compact algorithms with RI. Furthermore, Table 8 shows the results of the Holm-Bonferroni procedure applied to the eight algorithms and three problems.

Also in these experiments, RIcDE revealed superior (or at least equivalent) to all the other compact algorithms with and without RI. In particular, the pairwise Wilcoxon Rank-Sum test shows that RIcDE statistically outperforms all the compact algorithms without RI and is at least equivalent to (but, it is often better than) the other compact algorithms with RI. This further confirms the benefit obtained from the RI mechanism. On the other hand, according to the Holm-Bonferroni procedure, RIcDE is statistically equivalent to RIcGA, RIcBFO and (somehow surprisingly) cDE without RI: this might be due however to the limited number of problems (in this case, three) on which this analysis is performed. Still, the null-hypothesis is rejected in the case of cBFO, cGA and RIcPSO (which performs the worst, most likely because of the relatively low number of parameters of the three problems).

Table 6: Average fitness ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE vs. compact algorithms without RI on CEC2011 problems.

	RIcDE	cDE		cBFO		cGA		cPSO	
P_1	$\mathbf{4.68e+00 \pm 4.79e+00}$	$1.21e+01 \pm 7.65e+00$	+	$2.36e+01 \pm 8.90e+00$	+	$1.21e+01 \pm 7.71e+00$	+	$1.04e+01 \pm 5.28e+00$	+
P_2	$\mathbf{-2.40e+01 \pm 3.07e+00}$	$-2.03e+01 \pm 4.11e+00$	+	$-1.81e+01 \pm 2.96e+00$	+	$-8.32e+00 \pm 2.11e+00$	+	$-1.05e+01 \pm 1.58e+00$	+
P_7	$\mathbf{1.16e+00 \pm 1.63e-01}$	$1.30e+00 \pm 1.87e-01$	+	$1.32e+00 \pm 2.06e-01$	+	$1.36e+00 \pm 1.25e-01$	+	$1.66e+00 \pm 1.59e-01$	+

Table 7: Average fitness ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE vs. compact algorithms with RI on CEC2011 problems.

	RIcDE	RIcBFO		RIcGA		RIcPSO	
P_1	$\mathbf{4.68e+00 \pm 4.79e+00}$	$2.15e+01 \pm 5.70e+00$	+	$9.47e+00 \pm 5.65e+00$	+	$1.43e+01 \pm 3.43e+00$	+
P_2	$\mathbf{-2.40e+01 \pm 3.07e+00}$	$-1.94e+01 \pm 2.19e+00$	+	$-1.01e+01 \pm 1.27e+00$	+	$-8.53e+00 \pm 1.05e+00$	+
P_7	$1.16e+00 \pm 1.63e-01$	$1.17e+00 \pm 1.62e-01$	=	$\mathbf{1.16e+00 \pm 1.54e-01}$	=	$1.76e+00 \pm 1.02e-01$	+

Table 8: Holm-Bonferroni procedure (reference: RIcDE, Rank=7.67e+00) on CEC2011 problems.

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcGA	6.00e+00	-9.45e-01	1.72e-01	5.00e-02	Accepted
2	cDE	5.67e+00	-1.13e+00	1.28e-01	2.50e-02	Accepted
3	RIcBFO	4.67e+00	-1.70e+00	4.45e-02	1.67e-02	Accepted
4	cPSO	4.00e+00	-2.08e+00	1.88e-02	1.25e-02	Accepted
5	cBFO	3.33e+00	-2.46e+00	7.01e-03	1.00e-02	Rejected
6	cGA	2.67e+00	-2.83e+00	2.29e-03	8.33e-03	Rejected
7	RIcPSO	2.00e+00	-3.21e+00	6.57e-04	7.14e-03	Rejected

4.2. Parameter analysis

We conducted a thorough parameter analysis to show how the parametrization of the RI mechanism (α and local budget), as well as the kind of crossover employed (exponential or binomial), affects the algorithmic performance. The detailed results are reported in Appendix E.

Overall, the analysis shows that the exponential crossover produces in general better results, while the effect of the parameters seems somehow marginal, with slightly better results obtained for ($\alpha = 0.25$, local budget=30%). We should highlight though that for all the experiments reported in this paper we chose on purpose a different parametrization ($\alpha = 0.05$, local budget=25%), since 1) we preferred to avoid an excessive fine-tuning of the algorithms and

2) as shown in E.25 in the Appendix, the parametrization ($\alpha = 0.05$, local budget=25%) is actually the last one -in terms of Holm-Bonferroni Rank- to be considered statistically equivalent (null-hypothesis accepted) to the top-ranked setting ($\alpha = 0.25$, local budget=30%). Thus, using this one as "candidate" algorithm, rather than the top-ranked setting, provides a more conservative estimation of the algorithmic performance that is somehow less dependent on fine-tuned parameters.

4.3. Time and memory complexity

We conclude our analysis with some considerations on the time complexity of the RIcDE algorithm, as well as its required memory. We should remark that even though we focus the analysis on RIcDE, similar considerations hold true for the other compact algorithms considered in this study, either with or without RI.

First of all, we computed the *algorithmic overhead* that characterizes the original cDE algorithm and its RI counterpart (RIcDE), in comparison with the three single-solution algorithms and the three population-based algorithms that we tested on the CEC2014 and CEC2013-LSGO benchmarks (eight algorithms in total). We define the algorithmic overhead as the total time of an optimization run (until the end of the allotted budget of N_{evals} function evaluations), minus the total time needed to perform the N_{evals} function evaluations. In other words, this metric measures the actual time needed by the algorithm to perform its inner operations and thus conduct the search.

In order to estimate the overhead, we proceeded as follows. In the following, all the operations were performed in single thread. First, we collected the time needed to perform N_{evals} function evaluations of a given fitness function (here indicated with T_{evals}). In our experiments, we set $N_{evals} = 100000$ and considered the Sphere function $f(\mathbf{x}) = \sum_i x_i$. We repeated this measure 30 time to have a more robust estimation.

Then, we computed the time needed to run an optimization process consisting of N_{evals} function evaluations (here referred to as "elapsed time" $T_{elapsed}$).

Also in this case, we repeated this measure 30 times. Finally, we computed the difference $T_{elapsed} - T_{evals}$, averaged over the 30 performed repetitions to return the average algorithmic overhead values, expressed in ms, for the eight algorithms under consideration. To show the effect of increasing dimensional values on the average algorithmic overhead, the aforementioned process was repeated by scaling up the Sphere problem. The considered numbers of design variables were 10, 50, 100, 500 and 1000. The resulting average $T_{elapsed} - T_{evals}$ values, one for each algorithm and problem size, are depicted in Figure 4. It is worth highlighting that the bars in the figure are displayed in logarithmic scale to reduce the wide range induced by CMA-ES $(1 + 1)$, which has super-quadratic time complexity, to a more informative size.

A quantitative analysis (conducted with the Matlab curve fitting tool) confirmed that the time complexity for CLPSO, EPSDE-LS, JADE, nuSA is $\mathcal{O}(n)$ (the overhead data fit a linear curve, $R^2 = 1.0$); for PMS, cDE and RIcDE it is $\mathcal{O}(n^\alpha)$, $0 < \alpha < 1$ (respectively with: $\alpha = 0.6513$, $R^2 = 0.9997$; $\alpha = 0.9455$, $R^2 = 0.9999$; $\alpha = 0.8942$, $R^2 = 0.9996$); for CMA-ES (1+1), it is $\mathcal{O}(n^\alpha)$, $\alpha > 1$ ($\alpha = 2.205$, $R^2 = 1.0$). Thus we can conclude that the addition of the RI component to the original cDE scheme does not increase its time complexity. Further considerations on the number of iterations needed to converge are reported in Appendix F.

The second aspect we considered is memory. We compared the same eight algorithms in terms of (approximate) number of n-dimensional vectors stored in memory to allow the algorithmic operations, reported in Table 9, an approach similar to what was done in [33]. From the table, it can be seen that apart from nuSA, the cDE algorithm (with and without RI) requires significantly less memory than the population-based algorithms (which is expected), but also less memory than CMA-ES 1+1 and PMS that, despite conducting the search perturbing one solution at a time, need additional structures such as matrices to perform their operations.

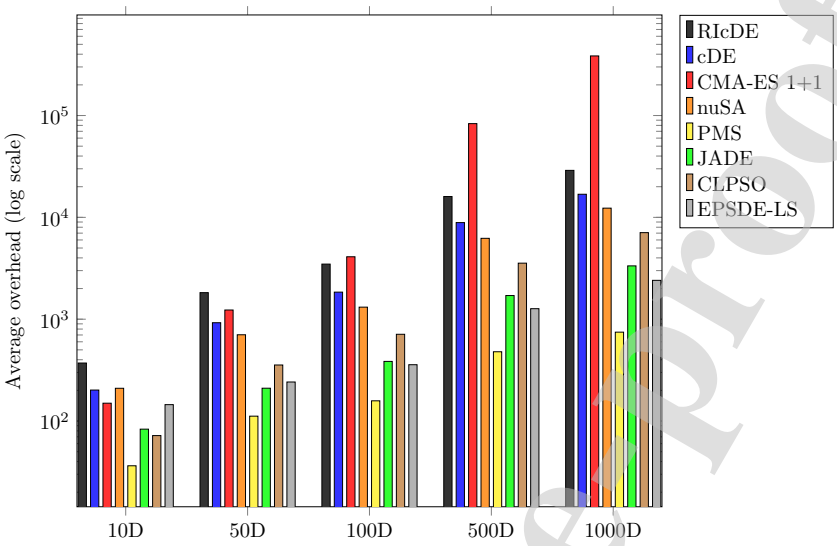

Figure 4: Average algorithmic overhead [ms] for the algorithms under study, over increasing dimensionality values.

Table 9: Memory complexity in terms of approximate number of "memory slots" (i.e. n-dimensional vectors) for the algorithms under study. For the population-based algorithms, N_p indicates the population size. Data taken from [38, 33].

Algorithm	**Memory slots**
cDE	4
RIcDE	4
nuSA	2
CMA-ES 1+1	n+2
PMS	$3 + n^2$
EPSDE-LS	$N_p + n^2$
CLPSO	$2 \times N_p$
JADE	$2 \times N_p$

5. Conclusions

In this paper, we have presented a simple framework for solving continuous optimization problems on devices characterized by limited memory. The proposed framework is based on a combination of a compact algorithm with

a restart mechanism based on Re-Sampled Inheritance (RI) that is meant to partially inherit some variables from the current best solution while initializing the others randomly. We tested the framework on four different compact algorithms presented in the literature (namely: cDE, cGA, cPSO, and cBFO) and performed numerical experiments on a broad range of benchmark functions taken from the CEC2011, CEC2013-LSGO and CEC2014 test suites, in several dimensionalities (from 10 to 1000).

Our experiments show that the use of RI consistently enhances the performances of compact algorithms, still keeping a limited usage of memory. Furthermore, among different restart mechanisms (e.g. based on the binomial crossover, as we tested in the additional experiments reported in the Appendix, or random restarts, tested in our previous work [42]), the exponential crossover seems to be the most successful one, as it guarantees a good compromise between a fully-disruptive random restart and an excessively "local" restart (with higher values of α, i.e., the ratio of inherited variables). Furthermore, RI based on the crossover operator (either binomial or exponential) appears very robust to its parametrization as for what concerns the local budget and the α parameter. In addition to that, we noted that among the tested algorithms the best performance was obtained by cDE with RI (RIcDE), especially up to mid-high dimensionalities (100D). On the other hand, when the problem dimensionality grows to very large scales (1000D), interestingly cPSO with RI (RIcPSO) performs much better than RIcDE and other compact algorithms. This somehow suggests that the PSO logics –even in its compact declination– is better suited for large scale problems, which is consistent with some of the recent trends on large scale optimization.

Another interesting finding is that, compared to state-of-the-art population-algorithms, RIcDE obtains competitive performances, being even superior to a few complex memory-hungry variants of Differential Evolution and Particle Swarm Optimization. However, in some cases the performance difference can be quite large in favour of the population-based algorithm. This is expected though: while the RI mechanism definitely helps solving some limitations of

compact algorithms, the lack of a population can still impede satisfactory performances on the most challenging optimization problems.This highlights the benefits of having a population of candidate solutions and suggests that compact optimization should be mainly used to optimize those systems presenting memory and time restrictions. Indeed, under such limitations, population-based algorithms would not be a suitable choice. Conversely, when memory is available and high-quality solutions are sought over multimodal and non-separable problems, population-based algorithms might still be preferable.

In future works, apart from extending the application of the RI mechanism to other Swarm Intelligence paradigms [59, 60] as well as Bayesian Optimization [61], we will follow on the findings of this study by investigating (self-)adaptive combinations of compact logics. According to our results, a hybrid scheme alternating cDE and cPSO perturbation logics would be particularly successful if the budget allocated to the two perturbation logics is adjusted based on the dimensionality of the problem (i.e. increased in favour of cPSO over large scale problems and vice versa). Additionally, even though the RI parametrization seems to have little influence on the overall algorithm performance, adapting the local budget on-the-go (and therefore the number of restarts) might also lead to an improvement upon the quality of the returned solution. As for the α parameter, a possible adaptation scheme might start with smaller values (to inherit fewer variables from the best solution, thus increasing exploration) and then increase it as the optimization process goes on (to inherit more variables, thus increasing exploitation). As a simpler alternative, the RI parametrization could also change depending on the problem dimensionality. It is worth stressing out that the employed compact algorithms can be also further modified to self-adapt their corresponding control parameters. Even though successful self-adaptation schemes based on population-diversity metrics, e.g. distances between candidate solutions [62, 63], cannot be easily translated to the compact domain, several other "randomized" parameters adaption schemes available in the literature, see e.g. [64, 65, 66, 67, 68, 69], would be logical choices to achieve self-adaptation in compact optimization.

References

[1] F. Neri, G. Iacca, E. Mininno, Compact optimization, in: Handbook of Optimization: From Classical to Modern Approach, Springer, 2013, pp. 337–364.

[2] P. Larrañaga, J. A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.

[3] Z. Ren, Y. Liang, L. Wang, A. Zhang, B. Pang, B. Li, Anisotropic adaptive variance scaling for Gaussian estimation of distribution algorithm, Knowl. Based Syst. 146 (2018) 142–151.

[4] Y. Sun, G. G. Yen, Z. Yi, Reference line-based estimation of distribution algorithm for many-objective optimization, Knowl. Based Syst. 132 (2017) 129–143.

[5] J. Madera, E. Alba, A. Ochoa, A parallel island model for estimation of distribution algorithms, in: Towards a New Evolutionary Computation, Springer, 2006, pp. 159–186.

[6] Y. Martínez-López, J. Madera, A. Y. Rodríguez-González, S. Barigye, Cellular estimation Gaussian algorithm for continuous domain, J. Intell. Fuzzy Syst. 36 (5) (2019) 4957–4967.

[7] G. R. Harik, F. G. Lobo, D. E. Goldberg, The compact genetic algorithm, IEEE T. Evolut. Comput. 3 (4) (1999) 287–297.

[8] F. Corno, M. S. Reorda, G. Squillero, The selfish gene algorithm: A new evolutionary optimization strategy, in: Symposium on Applied Computing, ACM, 1998, pp. 349–355.

[9] C. W. Ahn, R. S. Ramakrishna, Elitism-based compact genetic algorithms, IEEE T. Evolut. Comput. 7 (4) (2003) 367–385.

[10] J. C. Gallagher, S. Vigraham, G. Kramer, A family of compact genetic algorithms for intrinsic evolvable hardware, IEEE T. Evolut. Comput. 8 (2) (2004) 111–126.

[11] E. Mininno, F. Cupertino, D. Naso, Real-valued compact genetic algorithms for embedded microcontroller optimization, IEEE T. Evolut. Comput. 12 (2) (2008) 203–219.

[12] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact differential evolution, IEEE T. Evolut. Comput. 15 (1) (2011) 32–54.

[13] G. Iacca, R. Mallipeddi, E. Mininno, F. Neri, P. N. Suganthan, Super-fit and population size reduction in compact differential evolution, in: Workshop on Memetic Computing, IEEE, 2011, pp. 1–8.

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

[15] G. Iacca, E. Mininno, F. Neri, Composed compact differential evolution, Evol. Intell. 4 (1) (2011) 17–29.

[16] X. Xue, J. Liu, Collaborative ontology matching based on compact interactive evolutionary algorithm, Knowl. Based Syst. 137 (2017) 94–103.

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

[18] G. Iacca, F. Neri, E. Mininno, Compact bacterial foraging optimization, in: Swarm and Evolutionary Computation, 2012, pp. 84–92.

[19] Z. Yang, K. Li, Y. Guo, A new compact teaching-learning-based optimization method, in: International Conference on Intelligent Computing, 2014, pp. 717–726.

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

[21] A. Banitalebi, M. I. A. Aziz, A. Bahar, Z. A. Aziz, Enhanced compact artificial bee colony, Inform. Sciences 298 (2015) 491–511.

[22] T.-K. Dao, S.-C. Chu, C.-S. Shieh, M.-F. Horng, Compact artificial bee colony, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2014, pp. 96–105.

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

[24] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, S.-C. Chu, A compact artificial bee colony optimization for topology control scheme in wireless sensor networks, J. Inf. Hiding Multimedia Signal Process 6 (2) (2015) 297–310.

[25] G. Iacca, F. Caraffini, F. Neri, E. Mininno, Robot base disturbance optimization with compact differential evolution light, in: D. C. C. et al. (Ed.), Conference on the Applications of Evolutionary Computation, Springer, Berlin, Heidelberg, 2012, pp. 285–294.

[26] G. Iacca, F. Caraffini, F. Neri, Memory-saving memetic computing for path-following mobile robots, Appl. Soft Comput. 13 (4) (2013) 2003–2016.

[27] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing, Springer Berlin, 2007.

[28] A. Yaman, G. Iacca, F. Caraffini, A comparison of three differential evolution strategies in terms of early convergence with different population sizes, in: AIP Conference Proceedings, Vol. 2070-1, 2019, p. 020002.

[29] A. V. Kononova, D. W. Corne, P. D. Wilde, V. Shneer, F. Caraffini, Structural bias in population-based algorithms, Inform. Sciences 298 (2015) 468–490.

[30] F. Caraffini, A. V. Kononova, Structural bias in differential evolution: A preliminary study, in: AIP Conference Proceedings, Vol. 2070-1, 2019, p. 020005.

[31] A. Prugel-Bennett, Benefits of a population: Five mechanisms that advantage population-based algorithms, IEEE T. Evolut. Comput. 14 (4) (2010) 500–517.

[32] F. Caraffini, G. Iacca, F. Neri, L. Picinali, E. Mininno, A CMA-ES super-fit scheme for the re-sampled inheritance search, in: Congress on Evolutionary Computation, IEEE, 2013, pp. 1123–1130.

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

[34] F. Caraffini, G. Iacca, A. Yaman, Improving (1+1) covariance matrix adaptation evolution strategy: A simple yet efficient approach, in: AIP Conference Proceedings, Vol. 2070-1, 2019, p. 020004.

[35] K. V. Price, R. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, 2005.

[36] F. Caraffini, F. Neri, A study on rotation invariance in differential evolution, Swarm Evol. Comput. 50 (2019) 100436.

[37] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: Handbook of metaheuristics, Springer, 2003, pp. 320–353.

[38] G. Iacca, F. Neri, E. Mininno, Y.-S. Ong, M.-H. Lim, Ockham's razor in memetic computing: three stage optimal memetic exploration, Inform. Sciences 188 (2012) 17–43.

[39] F. Caraffini, F. Neri, M. Epitropakis, HyperSPAM: A study on hyper-heuristic coordination strategies in the continuous domain, Inform. Sciences 477 (2019) 186–202.

[40] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation CMA-ES, Evol. Comput. 11 (1) (2003) 1–18.

[41] C. Igel, T. Suttorp, N. Hansen, A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies, in: Genetic and Evolutionary Computation Conference, 2006, pp. 453–460.

[42] G. Iacca, F. Caraffini, Compact optimization algorithms with re-sampled inheritance, in: Conference on the Applications of Evolutionary Computation, Springer, 2019, pp. 523–534.

[43] J. J. Liang, B. Y. Qu, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Tech. rep., Zhengzhou University and Nanyang Technological University (2013).

[44] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, H. China, Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization, Tech. Rep. 33, RMIT University and University of Science and Technology of China and National University of Defense Technology (2013).

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

[46] W. Gautschi, Error function and Fresnel integrals, in: Handbook of mathematical functions, Vol. 55, Dover, 1972, pp. 297–308.

[47] W. J. Cody, Rational Chebyshev approximations for the error function, Mat. of Comp. 23 (107) (1969) 631–637.

[48] F. Caraffini, G. Iacca, The SOS platform: designing, tuning and statistically benchmarking optimisation algorithms, Mathematics 8 (5) (2020) 785.

[49] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2008) 959–977.

[50] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70.

[51] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.

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

[53] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inform. Sciences 227 (2013) 60–82.

[54] J. Zhang, A. C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE T. Evolut. Comput. 13 (5) (2009) 945–958.

[55] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE T. Evolut. Comput. 10 (3) (2006) 281–295.

[56] G. Iacca, F. Neri, F. Caraffini, P. N. Suganthan, A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms, in: Conference on the Applications of Evolutionary Computation, Springer, 2014, pp. 615–626.

[57] F. Neri, C. Cotta, P. Moscato, Handbook of memetic algorithms, Springer, 2012.

[58] G. Xue, R. S. Maier, J. B. Rosen, Minimizing the lennard-jones potential function on a massively parallel computer, in: Proceedings of the 6th international conference on Supercomputing, 1992, pp. 409–416.

[59] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, J. Global Optim. 39 (3) (2007) 459–471.

[60] S. Biswas, M. A. Eita, S. Das, A. V. Vasilakos, Evaluating the performance of group counseling optimizer on cec 2014 problems for computational expensive optimization, in: Congress on Evolutionary Computation, IEEE, 2014, pp. 1076–1083.

[61] M. Pelikan, D. E. Goldberg, E. Cantú-Paz, et al., Boa: The bayesian optimization algorithm, in: Genetic and Evolutionary Computation Conference, Vol. 1, Citeseer, 1999, pp. 525–532.

[62] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, A. Zamuda, Distance based parameter adaptation for success-history based differential evolution, Swarm Evol. Comput. 50 (2019) 100462.

[63] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, M. N. Vrahatis, Enhancing differential evolution utilizing proximity-based mutation operators, IEEE T. Evolut. Comput. 15 (1) (2011) 99–119.

[64] J. Brest, A. Zamuda, I. Fister, B. Boskovic, Some improvements of the self-adaptive jDE algorithm, in: Symposium on Differential Evolution, IEEE, 2014, pp. 1–8.

[65] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in differential evolution, Swarm Evol. Comput. 25 (2015) 72–99.

[66] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, A. Zamuda, Distance based parameter adaptation for success-history based differential evolution, Swarm Evol. Comput. 50 (2019) 100462.

[67] Y. Zhile, L. Kang, N. Qun, X. Yusheng, A. Foley, A self-learning TLBO based dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads, J. Mod. Power Syst. Cle. 2 (4) (2014) 298–307.

[68] J. Na, Z. Yang, S. Kamal, L. Hu, W. Wang, Y. Zhou, Bio-inspired learning and adaptation for optimization and control of complex systems, Complexity 2019.

[69] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Appl. Soft Comput. 27 (2015) 99–126.

Appendix A. Pseudo-code of the tested compact algorithms

We report here the pseudo-code of the compact algorithms used in our experimentation, namely cDE [14], cBFO [18], cGA [11], and cPSO [17]. We consider only the case where persistent elitism is used, see [1]. Furthermore, we include the pseudo-code for the "binomial" crossover operator, which we used in the parameter analysis process reported in Section Appendix E. A list of symbols used in the pseudo-codes is shown in Table A.10. In the pseudo-codes, all the binary and unary operations on vectors are assumed to be element-wise.

Symbol	Description
$\circ$	Element-wise product of vectors
$\mathbf{U}_1, \mathbf{U}_2$	Vector of random numbers sampled from $\mathcal{U}(0,1)$
u	Random number sampled from $\mathcal{U}(0,1)$
ϕ_1, ϕ_2, ϕ_3	Velocity update weights
γ_1, γ_2	Position update weights
C	Chemotactic step size
N_s	Swim steps
τ	Constant for elimination/dispersal

Table A.10: List of symbols used in the pseudo-codes of the compact algorithms.

Algorithm 4 Binomial crossover

Input: two parents $\mathbf{x}^A$ and $\mathbf{x}^B$ ▷ each one containing n variables
$\mathbf{x}^{offspring} = \mathbf{x}^A$
$i = 1$
while $i < n$ **do**
  if $r \sim \mathcal{U}(0,1) \leq Cr$ **then**
    $x_i^{offspring} = x_i^B$ ▷ exchange the i-th variable
  end if
  $i = i + 1$
end while
return $\mathbf{x}^{offspring}$

Algorithm 5 compact Differential Evolution – cDE

initialize **PV** ▷ $\mu_i = 0$, $\sigma_i = \lambda$ for each i-th variable
generate **elite** by means of **PV**
while stop condition is not met **do**
  compute $\mathbf{PV}' = \left[\boldsymbol{\mu}, (1 + 2F^2)\boldsymbol{\sigma}\right]$ ▷ mutation light
  generate a candidate solution $\mathbf{x}'$ by means of $\mathbf{PV}'$
  generate $\mathbf{x}$ solution through crossover on $\mathbf{x}'$ and **elite** ▷ Algorithm 3
  compare fitness $\mathbf{x}$ and **elite**
  update **PV** ▷ see Equations (2) and (3)
  if $\mathbf{x}$ outperforms **elite then**
    elite $= \mathbf{x}$
  end if
end while
return elite

Algorithm 6 compact Bacteria Foraging Optimization – cBFO

initialize **PV** ▷ $\mu_i = 0$, $\sigma_i = \lambda$ for each i-th variable
generate **elite** by means of **PV**
while stop condition is not met **do**
    generate a candidate solution $\mathbf{x}$ by means of **PV**
    compare fitness of $\mathbf{x}$ and **elite**
    update **PV** ▷ see Equations (2) and (3)
    generate a random $\boldsymbol{\Delta} \in [-1, 1]^n$ ▷ the "direction" vector
    $\mathbf{x}' = \mathbf{x} + C\boldsymbol{\Delta}/\sqrt{\boldsymbol{\Delta}^T\boldsymbol{\Delta}}$ ▷ tumble and move
    for $i = 1 : N_s$ **do** ▷ swim
        compare fitness of $\mathbf{x}'$ and **elite**
        update **PV** ▷ see Equations (2) and (3)
        if $\mathbf{x}'$ outperforms **elite then**
            elite $= \mathbf{x}$
        end if
        if $\mathbf{x}'$ outperforms $\mathbf{x}$ **then**
            $\mathbf{x}' = \mathbf{x}' + C\boldsymbol{\Delta}/\sqrt{\boldsymbol{\Delta}^T\boldsymbol{\Delta}}$ ▷ with same direction vector Δ
        end if
    end for
    compare fitness of $\boldsymbol{\mu}$ and **elite** ▷ reproduction: shift $\boldsymbol{\mu}$ towards **elite**
    update **PV**
    $\boldsymbol{\mu} = \boldsymbol{\mu} + 2\tau u - \tau$ ▷ elimination/dispersal: perturb **PV**
    $\boldsymbol{\sigma} = \sqrt{\boldsymbol{\sigma}^2 + \tau u}$
end while
return elite

Algorithm 7 compact Genetic Algorithm – cGA

initialize **PV** ▷ $\mu_i = 0$, $\sigma_i = \lambda$ for each i-th variable
generate **elite** by means of **PV**
while stop condition is not met **do**
    generate a candidate solution **x** by means of **PV**
    compare fitness of **x** and **elite**
    update **PV** ▷ see Equations (2) and (3)
    if x outperforms **elite then**
        elite = **x**
    end if
end while
return elite

Algorithm 8 compact Particle Swarm Optimization – cPSO

initialize **PV** ▷ $\mu_i = 0$, $\sigma_i = \lambda$ for each i-th variable
generate **elite** by means of **PV** ▷ the "global best" solution
generate **v** by means of **PV** ▷ the "velocity" vector
generate a candidate solution **x** by means of **PV**
while stop condition is not met **do**
    generate $\mathbf{x}^{lb}$ by means of **PV** ▷ the "local best" solution
    $\mathbf{v} = \phi_1\mathbf{v} + \phi_2\mathbf{U}_1 \circ \left(\mathbf{x}^{lb} - \mathbf{x}\right) + \phi_3\mathbf{U}_2 \circ (\mathbf{elite} - \mathbf{x})$ ▷ velocity update
    $\mathbf{x} = \gamma_1\mathbf{x} + \gamma_2\mathbf{v}$ ▷ position update
    compare fitness of **x** and $\mathbf{x}^{lb}$
    update **PV** ▷ see Equations (2) and (3)
    if x outperforms **elite then**
        elite = **x**
    end if
end while
return elite

Appendix B. Detailed results on CEC2014 and CEC2013-LSGO - Comparison against compact algorithms with and without RI

We report the detailed results of the RIcDE algorithm on the CEC2014 benchmark [43] in 10, 50 and 100 dimensions, and the CEC2013-LSGO benchmark [44] in 1000 dimensions, against the other compact algorithms with and without RI, as explained in Section 4.

In particular, Table B.11, B.12 and Table B.13 report the results of RIcDE and the other compact algorithms (without RI) on the CEC2014 problems in 10, 50 and 100 dimensions respectively. The tables show for each problem and algorithm the average error ± standard deviation across 30 independent runs and the result of the relative pairwise Wilcoxon Rank-Sum test. Similarly, Table B.14 reports the results of RIcDE and the other compact algorithms (without RI) on the CEC2013-LSGO benchmark in 1000 dimensions.

Finally, Table B.15, B.16 and Table B.17 report the results of RIcDE and the other compact algorithms (with RI) on the CEC2014 problems in 10, 50 and 100 dimensions respectively, while Table B.18 reports the results of RIcDE and the other compact algorithms (with RI) on the CEC2013-LSGO benchmark in 1000 dimensions.

Table B.11: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against compact algorithms (without RI) on CEC2014 in 10D.

	RIcDE	cDE		cBFO		cGA		cPSO	
f_1	$1.38e+05 \pm 1.55e+05$	$\mathbf{6.33e+04 \pm 5.38e+04}$	-	$3.08e+05 \pm 2.02e+05$	+	$4.50e+05 \pm 1.74e+06$	=	$2.36e+06 \pm 1.72e+06$	+
f_2	$\mathbf{8.96e+02 \pm 1.22e+03}$	$4.41e+03 \pm 3.13e+03$	+	$5.17e+04 \pm 2.49e+04$	+	$4.37e+03 \pm 3.50e+03$	+	$3.10e+07 \pm 1.34e+07$	+
f_3	$6.92e+02 \pm 6.26e+02$	$7.30e+03 \pm 6.04e+03$	+	$\mathbf{6.05e+01 \pm 3.60e+01}$	-	$8.60e+03 \pm 6.62e+03$	+	$5.67e+03 \pm 3.22e+03$	+
f_4	$\mathbf{2.76e+00 \pm 8.60e+00}$	$1.34e+01 \pm 1.63e+01$	=	$3.10e+01 \pm 1.16e+01$	+	$2.70e+01 \pm 1.42e+01$	+	$4.03e+01 \pm 1.71e+01$	+
f_5	$\mathbf{2.00e+01 \pm 3.23e-03}$	$2.00e+01 \pm 4.12e-05$	+	$2.02e+01 \pm 8.63e-02$	+	$2.00e+01 \pm 8.30e-05$	+	$2.02e+01 \pm 1.27e+00$	+
f_6	$\mathbf{1.16e+00 \pm 7.62e-01}$	$2.60e+00 \pm 1.46e+00$	+	$6.22e+00 \pm 8.30e-01$	+	$2.87e+00 \pm 1.76e+00$	+	$3.42e+00 \pm 1.40e+00$	+
f_7	$\mathbf{9.91e-02 \pm 4.62e-02}$	$1.72e-01 \pm 1.10e-01$	+	$4.58e-01 \pm 1.52e-01$	+	$1.77e-01 \pm 1.19e-01$	+	$1.65e+00 \pm 3.35e-01$	+
f_8	$\mathbf{6.63e-02 \pm 2.48e-01}$	$4.64e-01 \pm 6.15e-01$	=	$5.07e+01 \pm 5.79e+00$	+	$2.08e+01 \pm 9.06e+00$	+	$9.86e+00 \pm 5.13e+00$	+
f_9	$\mathbf{9.79e+00 \pm 3.44e+00}$	$1.83e+01 \pm 5.67e+00$	+	$4.93e+01 \pm 3.36e+00$	+	$2.15e+01 \pm 8.87e+00$	+	$3.25e+01 \pm 8.96e+00$	+
f_{10}	$\mathbf{3.84e+00 \pm 4.04e+00}$	$2.19e+01 \pm 3.82e+01$	+	$1.18e+03 \pm 1.95e+02$	+	$6.08e+02 \pm 2.90e+02$	+	$6.75e+01 \pm 5.14e+01$	+
f_{11}	$\mathbf{3.44e+02 \pm 1.62e+02}$	$5.01e+02 \pm 2.70e+02$	+	$1.10e+03 \pm 2.60e+02$	+	$8.17e+02 \pm 3.24e+02$	+	$5.34e+02 \pm 3.51e+02$	+
f_{12}	$\mathbf{6.47e-02 \pm 3.39e-02}$	$1.41e-01 \pm 1.17e-01$	+	$4.30e-01 \pm 2.47e-01$	+	$3.24e-01 \pm 3.02e-01$	+	$1.03e+00 \pm 3.87e-01$	+
f_{13}	$\mathbf{2.35e-01 \pm 8.82e-02}$	$3.08e-01 \pm 1.32e-01$	+	$3.42e-01 \pm 9.17e-02$	+	$2.90e-01 \pm 1.20e-01$	=	$5.02e-01 \pm 1.20e-01$	+
f_{14}	$2.41e-01 \pm 1.15e-01$	$4.25e-01 \pm 2.56e-01$	+	$\mathbf{2.36e-01 \pm 7.50e-02}$	=	$3.75e-01 \pm 1.96e-01$	+	$4.32e-01 \pm 1.92e-01$	+
f_{15}	$\mathbf{9.56e-01 \pm 3.66e-01}$	$1.95e+00 \pm 9.13e-01$	+	$2.15e+00 \pm 9.33e-01$	+	$1.51e+00 \pm 7.23e-01$	+	$5.13e+00 \pm 9.85e-01$	+
f_{16}	$\mathbf{2.16e+00 \pm 4.30e-01}$	$2.65e+00 \pm 5.80e-01$	+	$3.36e+00 \pm 2.91e-01$	+	$3.18e+00 \pm 3.86e-01$	+	$3.11e+00 \pm 3.36e-01$	+
f_{17}	$1.23e+04 \pm 1.73e+04$	$4.47e+04 \pm 6.13e+04$	+	$\mathbf{1.57e+03 \pm 8.93e+02}$	-	$8.67e+03 \pm 9.08e+03$	=	$1.20e+04 \pm 1.62e+04$	=
f_{18}	$\mathbf{2.96e+03 \pm 3.90e+03}$	$1.24e+04 \pm 1.17e+04$	+	$6.56e+03 \pm 1.38e+03$	+	$1.05e+04 \pm 1.15e+04$	+	$1.30e+04 \pm 1.03e+04$	+
f_{19}	$\mathbf{5.81e-01 \pm 4.08e-01}$	$8.57e-01 \pm 5.94e-01$	=	$3.48e+00 \pm 8.04e-01$	+	$3.06e+00 \pm 1.02e+00$	+	$3.26e+00 \pm 5.56e-01$	+
f_{20}	$8.91e+02 \pm 9.92e+02$	$6.36e+03 \pm 8.36e+03$	+	$\mathbf{6.39e+01 \pm 2.23e+01}$	-	$5.19e+03 \pm 6.32e+03$	+	$3.25e+03 \pm 4.29e+03$	+
f_{21}	$\mathbf{7.80e+02 \pm 1.08e+03}$	$8.11e+03 \pm 8.80e+03$	+	$2.51e+03 \pm 1.41e+03$	+	$5.92e+03 \pm 5.36e+03$	+	$2.88e+03 \pm 2.38e+03$	+
f_{22}	$\mathbf{4.64e+00 \pm 7.58e+00}$	$3.66e+01 \pm 5.34e+01$	+	$1.67e+02 \pm 1.98e+01$	+	$6.80e+01 \pm 6.39e+01$	+	$2.32e+01 \pm 7.36e+00$	+
f_{23}	$3.27e+02 \pm 9.21e+00$	$3.29e+02 \pm 1.18e-12$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.30e+02 \pm 1.08e+00$	=	$3.29e+02 \pm 9.93e-02$	+
f_{24}	$\mathbf{1.23e+02 \pm 6.41e+00}$	$1.36e+02 \pm 1.13e+01$	+	$2.00e+02 \pm 0.00e+00$	+	$1.30e+02 \pm 1.04e+01$	+	$1.42e+02 \pm 9.15e+00$	+
f_{25}	$\mathbf{1.64e+02 \pm 3.03e+01}$	$1.93e+02 \pm 2.06e+01$	+	$2.00e+02 \pm 1.28e+00$	+	$1.97e+02 \pm 1.54e+01$	+	$1.86e+02 \pm 2.25e+01$	+
f_{26}	$\mathbf{1.00e+02 \pm 5.99e-02}$	$1.00e+02 \pm 1.33e-01$	+	$1.80e+02 \pm 3.99e+01$	+	$1.00e+02 \pm 1.82e-01$	+	$1.00e+02 \pm 1.28e-01$	+
f_{27}	$\mathbf{1.37e+02 \pm 1.73e+02}$	$2.86e+02 \pm 1.83e+02$	+	$2.00e+02 \pm 0.00e+00$	=	$2.11e+02 \pm 1.93e+02$	=	$1.57e+02 \pm 1.91e+02$	+
f_{28}	$4.19e+02 \pm 4.55e+01$	$4.79e+02 \pm 5.77e+01$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$4.28e+02 \pm 8.09e+01$	=	$5.86e+02 \pm 6.51e+01$	+
f_{29}	$3.14e+02 \pm 5.70e+01$	$1.73e+05 \pm 5.17e+05$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$8.42e+02 \pm 6.55e+02$	+	$1.88e+04 \pm 1.92e+04$	+
f_{30}	$6.60e+02 \pm 1.55e+02$	$1.01e+03 \pm 3.53e+02$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$1.40e+03 \pm 6.64e+02$	+	$1.96e+03 \pm 6.34e+02$	+

Table B.12: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against compact algorithms (without RI) on CEC2014 in 50D.

	RIcDE	cDE		cBFO		cGA		cPSO	
f_1	$7.12e+06 \pm 2.83e+06$	$1.20e+07 \pm 4.83e+06$	+	$\mathbf{4.54e+06 \pm 1.22e+06}$	-	$1.02e+08 \pm 5.26e+07$	+	$4.56e+08 \pm 1.95e+08$	+
f_2	$\mathbf{1.67e+03 \pm 3.41e+03}$	$6.41e+03 \pm 8.41e+03$	+	$8.19e+05 \pm 2.76e+05$	+	$2.57e+09 \pm 1.55e+09$	+	$3.89e+10 \pm 6.45e+09$	+
f_3	$1.86e+04 \pm 7.37e+03$	$2.91e+04 \pm 1.00e+04$	+	$\mathbf{2.84e+03 \pm 6.59e+03}$	-	$1.80e+05 \pm 3.61e+04$	+	$1.27e+05 \pm 3.02e+04$	+
f_4	$\mathbf{1.24e+02 \pm 2.19e+01}$	$1.88e+02 \pm 5.29e+01$	+	$2.27e+02 \pm 3.92e+01$	+	$5.82e+02 \pm 2.98e+02$	+	$6.56e+03 \pm 1.63e+03$	+
f_5	$\mathbf{2.00e+01 \pm 9.92e-06}$	$2.00e+01 \pm 1.08e-02$	+	$2.06e+01 \pm 9.20e-02$	+	$2.03e+01 \pm 1.16e-01$	+	$2.12e+01 \pm 3.31e-02$	+
f_6	$\mathbf{3.25e+01 \pm 2.81e+00}$	$3.72e+01 \pm 4.20e+00$	+	$6.34e+01 \pm 3.86e+00$	+	$4.61e+01 \pm 5.47e+00$	+	$6.12e+01 \pm 3.22e+00$	+
f_7	$\mathbf{2.35e-02 \pm 1.61e-02}$	$6.31e-02 \pm 6.63e-02$	+	$8.31e-01 \pm 6.77e-02$	+	$3.09e+01 \pm 2.04e+01$	+	$3.75e+02 \pm 5.81e+01$	+
f_8	$\mathbf{4.09e+01 \pm 7.74e+00}$	$6.59e+01 \pm 1.36e+01$	+	$3.02e+02 \pm 1.18e+01$	+	$2.73e+02 \pm 7.29e+01$	+	$4.87e+02 \pm 2.69e+01$	+
f_9	$\mathbf{1.90e+02 \pm 2.94e+01}$	$2.58e+02 \pm 5.74e+01$	+	$4.12e+02 \pm 2.09e+01$	+	$3.51e+02 \pm 6.54e+01$	+	$5.80e+02 \pm 3.02e+01$	+
f_{10}	$\mathbf{9.08e+02 \pm 3.20e+02}$	$1.64e+03 \pm 4.53e+02$	+	$7.91e+03 \pm 4.78e+02$	+	$7.34e+03 \pm 9.08e+02$	+	$8.98e+03 \pm 1.33e+03$	+
f_{11}	$\mathbf{4.92e+03 \pm 6.44e+02}$	$5.74e+03 \pm 8.57e+02$	+	$8.36e+03 \pm 7.02e+02$	+	$7.98e+03 \pm 1.25e+03$	+	$1.33e+04 \pm 4.46e+02$	+
f_{12}	$\mathbf{2.05e-01 \pm 5.80e-02}$	$3.21e-01 \pm 1.16e-01$	+	$1.26e+00 \pm 4.88e-01$	+	$8.09e-01 \pm 4.45e-01$	+	$3.46e+00 \pm 2.98e-01$	+
f_{13}	$\mathbf{5.46e-01 \pm 7.83e-02}$	$6.93e-01 \pm 9.55e-02$	+	$5.46e-01 \pm 6.40e-02$	=	$6.08e-01 \pm 1.22e-01$	+	$4.20e+00 \pm 3.20e-01$	+
f_{14}	$3.38e-01 \pm 1.46e-01$	$5.51e-01 \pm 2.81e-01$	+	$\mathbf{2.88e-01} \pm 3.54e-02$	=	$5.62e+00 \pm 9.16e+00$	+	$8.79e+01 \pm 1.26e+01$	+
f_{15}	$\mathbf{2.81e+01 \pm 7.03e+00}$	$5.54e+01 \pm 2.79e+01$	+	$5.93e+01 \pm 1.14e+01$	+	$1.89e+04 \pm 3.54e+04$	+	$8.53e+04 \pm 4.31e+04$	+
f_{16}	$\mathbf{2.01e+01 \pm 3.84e-01}$	$2.12e+01 \pm 6.78e-01$	+	$2.26e+01 \pm 1.65e-01$	+	$2.24e+01 \pm 5.50e-01$	+	$2.22e+01 \pm 3.06e-01$	+
f_{17}	$1.21e+06 \pm 6.41e+05$	$2.41e+06 \pm 2.27e+06$	+	$\mathbf{3.39e+05 \pm 1.29e+05}$	-	$1.11e+07 \pm 7.49e+06$	+	$3.01e+07 \pm 1.49e+07$	+
f_{18}	$\mathbf{9.33e+02 \pm 1.24e+03}$	$2.03e+03 \pm 1.33e+03$	+	$3.29e+04 \pm 1.56e+04$	+	$2.99e+03 \pm 2.13e+03$	+	$1.84e+08 \pm 1.08e+08$	+
f_{19}	$\mathbf{3.43e+01 \pm 1.68e+01}$	$5.29e+01 \pm 2.24e+01$	+	$4.20e+01 \pm 2.51e+01$	=	$8.36e+01 \pm 2.14e+01$	+	$1.80e+02 \pm 3.48e+01$	+
f_{20}	$2.95e+04 \pm 1.19e+04$	$4.30e+04 \pm 1.84e+04$	+	$\mathbf{5.87e+02 \pm 1.80e+02}$	-	$1.06e+05 \pm 5.79e+04$	+	$3.53e+04 \pm 1.98e+04$	=
f_{21}	$6.52e+05 \pm 5.03e+05$	$1.32e+06 \pm 1.14e+06$	+	$2.37e+\mathbf{05 \pm 9.93e+04}$	-	$6.57e+06 \pm 4.29e+06$	+	$8.51e+06 \pm 5.10e+06$	+
f_{22}	$\mathbf{9.37e+02 \pm 2.52e+02}$	$1.26e+03 \pm 2.79e+02$	+	$1.67e+03 \pm 3.48e+02$	+	$1.11e+03 \pm 2.92e+02$	+	$2.26e+03 \pm 2.04e+02$	+
f_{23}	$3.41e+02 \pm 6.99e-01$	$3.44e+02 \pm 2.74e+00$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.81e+02 \pm 2.73e+01$	+	$4.31e+02 \pm 1.38e+01$	+
f_{24}	$2.76e+02 \pm 4.60e+00$	$2.82e+02 \pm 4.11e+00$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.33e+02 \pm 1.40e+01$	+	$3.72e+02 \pm 1.15e+01$	+
f_{25}	$2.15e+02 \pm 5.63e+00$	$2.24e+02 \pm 4.11e+00$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$2.52e+02 \pm 1.28e+01$	+	$2.84e+02 \pm 2.03e+01$	+
f_{26}	$\mathbf{1.24e+02 \pm 4.23e+01}$	$1.65e+02 \pm 6.50e+01$	+	$2.00e+02 \pm 0.00e+00$	+	$1.62e+02 \pm 5.32e+01$	+	$1.37e+02 \pm 5.72e+01$	+
f_{27}	$1.19e+03 \pm 7.85e+01$	$1.36e+03 \pm 1.13e+02$	+	$\mathbf{2.00e+02 \pm 1.36e-12}$	-	$1.44e+03 \pm 1.54e+02$	+	$2.10e+03 \pm 1.93e+02$	+
f_{28}	$2.03e+03 \pm 5.88e+02$	$2.60e+03 \pm 8.62e+02$	+	$\mathbf{2.00e+02 \pm 1.36e-12}$	-	$2.55e+03 \pm 6.89e+02$	+	$7.87e+03 \pm 1.41e+03$	+
f_{29}	$1.67e+03 \pm 5.49e+02$	$2.43e+03 \pm 1.77e+03$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$2.00e+04 \pm 1.50e+04$	+	$4.35e+08 \pm 1.57e+08$	+
f_{30}	$1.16e+04 \pm 1.38e+03$	$1.50e+04 \pm 3.16e+03$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$7.66e+04 \pm 5.32e+04$	+	$1.83e+06 \pm 2.17e+06$	+

Table B.13: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against compact algorithms (without RI) on CEC2014 in 100D.

	RIcDE	cDE		cBFO		cGA		cPSO	
f_1	$8.59e+07 \pm 2.56e+07$	$1.72e+08 \pm 5.11e+07$	+	$\mathbf{6.46e+07 \pm 1.54e+07}$	-	$8.10e+08 \pm 1.88e+08$	+	$2.53e+09 \pm 6.99e+08$	+
f_2	$4.40e+07 \pm 3.21e+07$	$7.45e+08 \pm 4.18e+08$	+	$\mathbf{3.13e+06 \pm 5.78e+05}$	-	$5.35e+10 \pm 1.99e+10$	+	$1.80e+11 \pm 1.31e+10$	+
f_3	$3.68e+04 \pm 1.36e+04$	$5.74e+04 \pm 1.55e+04$	+	$\mathbf{2.22e+04 \pm 1.53e+04}$	-	$4.83e+05 \pm 8.37e+04$	+	$3.67e+05 \pm 4.10e+04$	+
f_4	$\mathbf{4.44e+02 \pm 6.92e+01}$	$7.01e+02 \pm 1.21e+02$	+	$5.06e+02 \pm 4.08e+01$	+	$6.43e+03 \pm 2.50e+03$	+	$3.36e+04 \pm 4.14e+03$	+
f_5	$\mathbf{2.00e+01 \pm 2.54e-02}$	$2.01e+01 \pm 5.13e-02$	+	$2.09e+01 \pm 4.80e-02$	+	$2.07e+01 \pm 1.05e-01$	+	$2.13e+01 \pm 1.95e-02$	+
f_6	$\mathbf{9.57e+01 \pm 5.43e+00}$	$1.02e+02 \pm 7.54e+00$	+	$1.46e+02 \pm 4.77e+00$	+	$1.14e+02 \pm 8.21e+00$	+	$1.48e+02 \pm 5.77e+00$	+
f_7	$\mathbf{1.34e+00 \pm 5.34e-01}$	$8.77e+00 \pm 6.07e+00$	+	$1.41e+00 \pm 2.35e-01$	=	$5.34e+02 \pm 1.31e+02$	+	$1.68e+03 \pm 1.38e+02$	+
f_8	$\mathbf{2.03e+02 \pm 2.51e+01}$	$2.62e+02 \pm 3.86e+01$	+	$6.02e+02 \pm 1.80e+01$	+	$9.11e+02 \pm 1.46e+02$	+	$1.27e+03 \pm 4.69e+01$	+
f_9	$\mathbf{6.04e+02 \pm 6.31e+01}$	$7.37e+02 \pm 1.11e+02$	+	$7.93e+02 \pm 1.58e+01$	+	$1.20e+03 \pm 1.54e+02$	+	$1.41e+03 \pm 7.19e+01$	+
f_{10}	$\mathbf{4.83e+03 \pm 9.31e+02}$	$6.83e+03 \pm 8.95e+02$	+	$1.67e+04 \pm 5.54e+02$	+	$1.93e+04 \pm 2.10e+03$	+	$2.57e+04 \pm 9.83e+02$	+
f_{11}	$\mathbf{1.33e+04 \pm 8.56e+02}$	$1.48e+04 \pm 1.39e+03$	+	$1.38e+04 \pm 6.78e+02$	+	$2.12e+04 \pm 1.48e+03$	+	$3.06e+04 \pm 5.08e+02$	+
f_{12}	$\mathbf{5.29e-01 \pm 8.81e-02}$	$6.93e-01 \pm 1.29e-01$	+	$2.05e+00 \pm 5.22e-01$	+	$1.24e+00 \pm 4.94e-01$	+	$4.21e+00 \pm 1.91e-01$	+
f_{13}	$6.23e-01 \pm 6.06e-02$	$7.01e-01 \pm 6.27e-02$	+	$\mathbf{5.18e-01 \pm 5.24e-02}$	-	$3.49e+00 \pm 5.71e-01$	+	$7.01e+00 \pm 2.42e-01$	+
f_{14}	$4.00e-01 \pm 1.56e-01$	$5.22e-01 \pm 2.31e-01$	+	$\mathbf{3.14e-01 \pm 2.46e-02}$	-	$1.32e+02 \pm 3.72e+01$	+	$4.97e+02 \pm 3.89e+01$	+
f_{15}	$4.35e+02 \pm 2.21e+02$	$1.97e+03 \pm 1.11e+03$	+	$\mathbf{1.75e+02 \pm 2.53e+01}$	-	$1.16e+06 \pm 7.74e+05$	+	$4.96e+06 \pm 2.08e+06$	+
f_{16}	$\mathbf{4.38e+01 \pm 9.04e-01}$	$4.47e+01 \pm 7.18e-01$	+	$4.61e+01 \pm 2.33e-01$	+	$4.59e+01 \pm 6.46e-01$	+	$4.64e+01 \pm 2.15e-01$	+
f_{17}	$9.59e+06 \pm 4.52e+06$	$1.86e+07 \pm 8.47e+06$	+	$\mathbf{2.13e+06 \pm 6.79e+05}$	-	$1.03e+08 \pm 4.33e+07$	+	$2.71e+08 \pm 1.68e+08$	+
f_{18}	$\mathbf{3.45e+03 \pm 4.04e+03}$	$7.46e+05 \pm 3.06e+06$	+	$1.14e+05 \pm 1.87e+04$	+	$1.63e+07 \pm 4.13e+07$	+	$2.94e+09 \pm 7.30e+08$	+
f_{19}	$\mathbf{1.20e+02 \pm 1.69e+01}$	$1.67e+02 \pm 3.91e+01$	+	$1.54e+02 \pm 3.11e+01$	+	$3.25e+02 \pm 9.91e+01$	+	$1.20e+03 \pm 2.17e+02$	+
f_{20}	$9.02e+04 \pm 2.84e+04$	$1.08e+05 \pm 3.89e+04$	=	$\mathbf{1.73e+03 \pm 8.73e+02}$	-	$4.23e+05 \pm 1.96e+05$	+	$2.59e+05 \pm 1.14e+05$	+
f_{21}	$4.70e+06 \pm 1.79e+06$	$8.82e+06 \pm 4.25e+06$	+	$\mathbf{2.58e+06 \pm 6.20e+05}$	-	$4.69e+07 \pm 1.76e+07$	+	$1.05e+08 \pm 5.13e+07$	+
f_{22}	$\mathbf{2.24e+03 \pm 3.44e+02}$	$2.79e+03 \pm 6.02e+02$	+	$3.15e+03 \pm 4.90e+02$	+	$2.89e+03 \pm 5.37e+02$	+	$5.72e+03 \pm 3.26e+02$	+
f_{23}	$3.42e+02 \pm 2.73e+00$	$4.16e+02 \pm 3.26e+01$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$6.08e+02 \pm 8.90e+01$	+	$7.54e+02 \pm 5.90e+01$	+
f_{24}	$4.08e+02 \pm 2.66e+01$	$4.44e+02 \pm 9.24e+00$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$7.62e+02 \pm 6.26e+01$	+	$7.54e+02 \pm 3.50e+01$	+
f_{25}	$2.67e+02 \pm 1.89e+01$	$3.02e+02 \pm 1.94e+01$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.89e+02 \pm 3.49e+01$	+	$5.72e+02 \pm 6.83e+01$	+
f_{26}	$2.05e+02 \pm 2.09e+00$	$2.07e+02 \pm 2.00e+01$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$2.56e+02 \pm 1.53e+01$	+	$3.93e+02 \pm 6.08e+01$	+
f_{27}	$2.76e+03 \pm 1.30e+02$	$3.01e+03 \pm 2.29e+02$	+	$\mathbf{2.00e+02 \pm 2.27e-12}$	-	$3.20e+03 \pm 1.89e+02$	+	$4.65e+03 \pm 1.61e+02$	+
f_{28}	$6.15e+03 \pm 1.24e+03$	$8.12e+03 \pm 1.82e+03$	+	$\mathbf{2.00e+02 \pm 2.27e-12}$	-	$6.56e+03 \pm 1.19e+03$	=	$2.33e+04 \pm 2.00e+03$	+
f_{29}	$4.99e+03 \pm 1.08e+03$	$9.79e+03 \pm 3.93e+03$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$7.73e+06 \pm 1.47e+07$	+	$2.14e+09 \pm 5.18e+08$	+
f_{30}	$5.30e+04 \pm 2.34e+04$	$1.14e+05 \pm 5.16e+04$	+	$2.00e+02 \pm 0.00e+00$	-	$2.29e+06 \pm 1.01e+06$	+	$1.48e+07 \pm 1.57e+07$	+

Table B.14: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against compact algorithms (without RI) on CEC2013-LSGO in 1000D.

	RIcDE	cDE		cBFO		cGA		cPSO	
f_1	$1.75e+11 \pm 1.13e+09$	$1.77e+11 \pm 1.46e+09$	+	$1.40e+11 \pm 7.60e+08$	-	$2.01e+11 \pm 1.78e+09$	+	$\mathbf{1.11e+11 \pm 6.76e+08}$	-
f_2	$1.88e+11 \pm 1.37e+08$	$1.88e+11 \pm 1.26e+08$	=	$2.06e+11 \pm 2.01e+08$	+	$\mathbf{1.86e+11 \pm 1.11e+08}$	-	$1.94e+11 \pm 5.96e+08$	+
f_3	$1.09e+11 \pm 3.42e+08$	$1.41e+11 \pm 7.99e+08$	+	$2.03e+11 \pm 1.04e+09$	+	$\mathbf{9.50e+10 \pm 2.15e+08}$	-	$1.53e+11 \pm 1.99e+09$	+
f_4	$1.74e+11 \pm 1.01e+09$	$2.19e+11 \pm 2.09e+09$	+	$1.93e+11 \pm 1.15e+09$	+	$1.77e+11 \pm 1.28e+09$	+	$\mathbf{1.15e+11 \pm 2.02e+09}$	-
f_5	$\mathbf{1.85e+11 \pm 1.71e+08}$	$1.87e+11 \pm 7.95e+07$	+	$2.05e+11 \pm 2.64e+08$	+	$1.91e+11 \pm 1.45e+08$	+	$1.93e+11 \pm 3.61e+08$	+
f_6	$1.21e+11 \pm 6.63e+08$	$1.23e+11 \pm 5.45e+08$	+	$2.00e+11 \pm 6.20e+08$	+	$\mathbf{1.13e+11 \pm 6.52e+08}$	-	$1.53e+11 \pm 1.80e+09$	+
f_7	$1.63e+11 \pm 6.64e+08$	$1.83e+11 \pm 1.57e+09$	+	$1.29e+11 \pm 6.64e+08$	-	$2.05e+11 \pm 3.46e+09$	+	$\mathbf{1.16e+11 \pm 1.05e+09}$	-
f_8	$2.10e+11 \pm 1.92e+09$	$2.39e+11 \pm 1.80e+09$	+	$1.94e+11 \pm 9.31e+08$	-	$1.56e+11 \pm 6.93e+08$	-	$\mathbf{1.09e+11 \pm 6.74e+08}$	-
f_9	$1.64e+11 \pm 3.54e+09$	$1.80e+11 \pm 1.64e+09$	+	$1.94e+11 \pm 5.69e+08$	+	$1.98e+11 \pm 1.18e+09$	+	$\mathbf{1.17e+11 \pm 1.78e+09}$	-
f_{10}	$1.35e+11 \pm 7.85e+08$	$1.37e+11 \pm 1.19e+09$	+	$2.03e+11 \pm 4.40e+08$	+	$\mathbf{1.12e+11 \pm 3.91e+08}$	-	$1.56e+11 \pm 2.35e+09$	+
f_{11}	$1.98e+11 \pm 2.73e+09$	$2.52e+11 \pm 2.13e+09$	+	$1.94e+11 \pm 9.89e+08$	-	$1.65e+11 \pm 1.23e+09$	-	$\mathbf{1.15e+11 \pm 8.99e+08}$	-
f_{12}	$2.06e+11 \pm 1.34e+09$	$2.11e+11 \pm 2.16e+09$	+	$1.46e+11 \pm 1.09e+09$	-	$1.79e+11 \pm 1.57e+09$	-	$\mathbf{1.21e+11 \pm 1.63e+09}$	-
f_{13}	$2.01e+11 \pm 2.33e+09$	$2.43e+11 \pm 1.67e+09$	+	$1.93e+11 \pm 1.01e+09$	-	$2.06e+11 \pm 3.38e+09$	+	$\mathbf{1.16e+11 \pm 1.17e+09}$	-
f_{14}	$1.83e+11 \pm 1.69e+09$	$1.87e+11 \pm 1.54e+09$	+	$1.95e+11 \pm 9.97e+08$	+	$1.78e+11 \pm 1.67e+09$	-	$\mathbf{1.20e+11 \pm 1.92e+09}$	-
f_{15}	$2.10e+11 \pm 1.68e+09$	$1.67e+11 \pm 2.37e+09$	-	$1.91e+11 \pm 8.60e+08$	-	$1.84e+11 \pm 2.31e+09$	-	$\mathbf{1.18e+11 \pm 9.47e+08}$	-

Table B.15: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against the other compact algorithms (with RI) on CEC2014 in 10D.

	RIcDE	RIcBFO		RIcGA		RIcPSO	
f_1	$\mathbf{1.38e+05 \pm 1.55e+05}$	$8.83e+05 \pm 3.54e+05$	+	$1.38e+05 \pm 7.91e+04$	=	$2.78e+06 \pm 1.48e+06$	+
f_2	$\mathbf{8.96e+02 \pm 1.22e+03}$	$6.23e+05 \pm 3.39e+05$	+	$1.06e+03 \pm 1.58e+03$	=	$4.08e+07 \pm 1.69e+07$	+
f_3	$\mathbf{6.92e+02 \pm 6.26e+02}$	$1.14e+03 \pm 3.62e+02$	+	$3.71e+03 \pm 2.81e+03$	+	$7.23e+03 \pm 2.87e+03$	+
f_4	$\mathbf{2.76e+00 \pm 8.60e+00}$	$1.99e+01 \pm 1.54e+01$	+	$1.67e+01 \pm 1.55e+01$	+	$4.44e+01 \pm 1.45e+01$	+
f_5	$\mathbf{2.00e+01 \pm 3.23e-03}$	$2.03e+01 \pm 7.69e-02$	+	$2.00e+01 \pm 8.08e-04$	+	$2.02e+01 \pm 1.38e+00$	+
f_6	$1.16e+00 \pm 7.62e-01$	$5.88e+00 \pm 6.44e-01$	+	$\mathbf{1.14e+00 \pm 7.46e-01}$	=	$3.45e+00 \pm 9.73e-01$	+
f_7	$9.91e-02 \pm 4.62e-02$	$7.24e-01 \pm 1.16e-01$	+	$\mathbf{8.69e-02 \pm 3.11e-02}$	=	$1.84e+00 \pm 2.98e-01$	+
f_8	$\mathbf{6.63e-02 \pm 2.48e-01}$	$4.58e+01 \pm 4.96e+00$	+	$1.02e+01 \pm 3.63e+00$	+	$1.49e+01 \pm 6.07e+00$	+
f_9	$\mathbf{9.79e+00 \pm 3.44e+00}$	$4.50e+01 \pm 5.12e+00$	+	$1.28e+01 \pm 4.25e+00$	+	$3.58e+01 \pm 8.16e+00$	+
f_{10}	$\mathbf{3.84e+00 \pm 4.04e+00}$	$1.02e+03 \pm 1.69e+02$	+	$3.07e+02 \pm 1.42e+02$	+	$3.06e+02 \pm 1.42e+02$	+
f_{11}	$\mathbf{3.44e+02 \pm 1.62e+02}$	$9.45e+02 \pm 1.25e+02$	+	$4.33e+02 \pm 1.86e+02$	=	$9.02e+02 \pm 3.07e+02$	+
f_{12}	$\mathbf{6.47e-02 \pm 3.39e-02}$	$5.32e-01 \pm 1.85e-01$	+	$1.09e-01 \pm 7.83e-02$	+	$1.31e+00 \pm 2.20e-01$	+
f_{13}	$2.35e-01 \pm 8.82e-02$	$3.04e-01 \pm 6.94e-02$	+	$\mathbf{1.77e-01 \pm 5.05e-02}$	-	$4.65e-01 \pm 9.36e-02$	+
f_{14}	$2.41e-01 \pm 1.15e-01$	$\mathbf{2.26e-01 \pm 4.85e-02}$	=	$2.28e-01 \pm 5.64e-02$	=	$3.74e-01 \pm 7.24e-02$	+
f_{15}	$9.56e-01 \pm 3.66e-01$	$2.15e+00 \pm 5.23e-01$	+	$\mathbf{7.42e-01 \pm 2.68e-01}$	-	$5.31e+00 \pm 7.10e-01$	+
f_{16}	$\mathbf{2.16e+00 \pm 4.30e-01}$	$3.14e+00 \pm 2.41e-01$	+	$2.73e+00 \pm 3.51e-01$	+	$3.22e+00 \pm 2.40e-01$	+
f_{17}	$1.23e+04 \pm 1.73e+04$	$\mathbf{1.48e+03 \pm 4.32e+02}$	-	$2.75e+03 \pm 2.60e+03$	-	$7.94e+03 \pm 6.45e+03$	=
f_{18}	$2.96e+03 \pm 3.90e+03$	$5.55e+03 \pm 1.06e+03$	+	$\mathbf{1.94e+03 \pm 4.32e+03}$	=	$3.71e+03 \pm 2.86e+03$	+
f_{19}	$\mathbf{5.81e-01 \pm 4.08e-01}$	$3.19e+00 \pm 5.23e-01$	+	$2.13e+00 \pm 5.37e-01$	+	$3.72e+00 \pm 4.34e-01$	+
f_{20}	$8.91e+02 \pm 9.92e+02$	$\mathbf{2.11e+02 \pm 1.62e+02}$	=	$6.19e+02 \pm 9.17e+02$	=	$1.02e+03 \pm 1.09e+03$	=
f_{21}	$\mathbf{7.80e+02 \pm 1.08e+03}$	$1.82e+03 \pm 9.12e+02$	+	$1.41e+03 \pm 1.87e+03$	+	$2.43e+03 \pm 1.26e+03$	+
f_{22}	$\mathbf{4.64e+00 \pm 7.58e+00}$	$1.55e+02 \pm 6.94e+00$	+	$2.96e+01 \pm 1.20e+01$	+	$3.01e+01 \pm 3.34e+00$	+
f_{23}	$3.27e+02 \pm 9.21e+00$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.27e+02 \pm 1.08e+01$	+	$3.30e+02 \pm 2.31e-01$	+
f_{24}	$1.23e+02 \pm 6.41e+00$	$2.00e+02 \pm 0.00e+00$	+	$\mathbf{1.22e+02 \pm 5.36e+00}$	=	$1.40e+02 \pm 8.65e+00$	+
f_{25}	$\mathbf{1.64e+02 \pm 3.03e+01}$	$2.00e+02 \pm 0.00e+00$	+	$1.73e+02 \pm 3.01e+01$	=	$1.69e+02 \pm 1.65e+01$	=
f_{26}	$1.00e+02 \pm 5.99e-02$	$1.49e+02 \pm 4.79e+01$	+	$\mathbf{1.00e+02 \pm 6.73e-02}$	=	$1.00e+02 \pm 1.08e-01$	+
f_{27}	$1.37e+02 \pm 1.73e+02$	$2.00e+02 \pm 0.00e+00$	=	$\mathbf{4.45e+01 \pm 1.19e+02}$	-	$5.40e+01 \pm 1.17e+02$	-
f_{28}	$4.19e+02 \pm 4.55e+01$	$2.00e+\mathbf{02 \pm 0.00e+00}$	-	$3.69e+02 \pm 4.58e+00$	-	$5.33e+02 \pm 5.01e+01$	+
f_{29}	$3.14e+02 \pm 5.70e+01$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$4.87e+02 \pm 1.72e+02$	+	$1.33e+04 \pm 1.45e+04$	+
f_{30}	$6.60e+02 \pm 1.55e+02$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$7.61e+02 \pm 1.92e+02$	+	$1.88e+03 \pm 5.49e+02$	+

Table B.16: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against the other compact algorithms (with RI) on CEC2014 in 50D.

	RIcDE	RIcBFO		RIcGA		RIcPSO	
f_1	$\mathbf{7.12e+06 \pm 2.83e+06}$	$1.04e+07 \pm 1.83e+06$	+	$5.60e+07 \pm 2.15e+07$	+	$4.37e+08 \pm 1.40e+08$	+
f_2	$\mathbf{1.67e+03 \pm 3.41e+03}$	$6.20e+06 \pm 1.01e+06$	+	$1.16e+09 \pm 4.77e+08$	+	$4.53e+10 \pm 4.65e+09$	+
f_3	$1.86e+04 \pm 7.37e+03$	$\mathbf{9.09e+03 \pm 3.98e+03}$	-	$1.30e+05 \pm 2.86e+04$	+	$1.27e+05 \pm 3.39e+04$	+
f_4	$\mathbf{1.24e+02 \pm 2.19e+01}$	$1.91e+02 \pm 3.12e+01$	+	$4.02e+02 \pm 9.70e+01$	+	$7.13e+03 \pm 1.40e+03$	+
f_5	$\mathbf{2.00e+01 \pm 9.92e-06}$	$2.07e+01 \pm 8.04e-02$	+	$2.02e+01 \pm 7.59e-02$	+	$2.11e+01 \pm 4.49e-02$	+
f_6	$\mathbf{3.25e+01 \pm 2.81e+00}$	$6.11e+01 \pm 2.29e+00$	+	$3.99e+01 \pm 3.98e+00$	+	$6.11e+01 \pm 2.36e+00$	+
f_7	$\mathbf{2.35e-02 \pm 1.61e-02}$	$8.95e-01 \pm 3.81e-02$	+	$1.45e+01 \pm 8.61e+00$	+	$4.17e+02 \pm 4.23e+01$	+
f_8	$\mathbf{4.09e+01 \pm 7.74e+00}$	$2.89e+02 \pm 9.64e+00$	+	$2.19e+02 \pm 3.18e+01$	+	$4.94e+02 \pm 3.11e+01$	+
f_9	$\mathbf{1.90e+02 \pm 2.94e+01}$	$3.97e+02 \pm 1.45e+01$	+	$3.01e+02 \pm 4.01e+01$	+	$5.82e+02 \pm 2.94e+01$	+
f_{10}	$\mathbf{9.08e+02 \pm 3.20e+02}$	$7.41e+03 \pm 4.35e+02$	+	$6.40e+03 \pm 7.07e+02$	+	$8.67e+03 \pm 8.49e+02$	+
f_{11}	$\mathbf{4.92e+03 \pm 6.44e+02}$	$8.12e+03 \pm 5.96e+02$	+	$6.84e+03 \pm 6.73e+02$	+	$1.32e+04 \pm 4.90e+02$	+
f_{12}	$\mathbf{2.05e-01 \pm 5.80e-02}$	$9.30e-01 \pm 2.43e-01$	+	$4.56e-01 \pm 1.46e-01$	+	$3.61e+00 \pm 2.45e-01$	+
f_{13}	$5.46e-01 \pm 7.83e-02$	$5.04e-01 \pm 4.72e-02$	-	$\mathbf{4.49e-01 \pm 8.09e-02}$	-	$4.32e+00 \pm 2.51e-01$	+
f_{14}	$3.38e-01 \pm 1.46e-01$	$\mathbf{2.66e-01 \pm 2.17e-02}$	-	$4.70e-01 \pm 4.03e-01$	+	$1.05e+02 \pm 1.13e+01$	+
f_{15}	$\mathbf{2.81e+01 \pm 7.03e+00}$	$5.36e+01 \pm 7.22e+00$	+	$2.07e+03 \pm 1.93e+03$	+	$1.26e+05 \pm 6.33e+04$	+
f_{16}	$\mathbf{2.01e+01 \pm 3.84e-01}$	$2.22e+01 \pm 1.86e-01$	+	$2.16e+01 \pm 5.94e-01$	+	$2.23e+01 \pm 2.42e-01$	+
f_{17}	$1.21e+06 \pm 6.41e+05$	$\mathbf{5.36e+05 \pm 1.69e+05}$	-	$5.63e+06 \pm 2.96e+06$	+	$2.49e+07 \pm 1.46e+07$	+
f_{18}	$\mathbf{9.33e+02 \pm 1.24e+03}$	$8.46e+04 \pm 2.64e+04$	+	$9.61e+02 \pm 3.90e+02$	+	$2.19e+08 \pm 7.36e+07$	+
f_{19}	$3.43e+01 \pm 1.68e+01$	$\mathbf{2.54e+01 \pm 2.01e+00}$	=	$5.82e+01 \pm 1.28e+01$	+	$1.97e+02 \pm 2.46e+01$	+
f_{20}	$2.95e+04 \pm 1.19e+04$	$\mathbf{1.00e+03 \pm 3.16e+02}$	-	$5.46e+04 \pm 1.87e+04$	+	$4.05e+04 \pm 1.34e+04$	+
f_{21}	$6.52e+05 \pm 5.03e+05$	$\mathbf{4.53e+05 \pm 8.03e+04}$	=	$2.73e+06 \pm 1.41e+06$	+	$6.32e+06 \pm 3.33e+06$	+
f_{22}	$9.37e+02 \pm 2.52e+02$	$1.37e+03 \pm 1.94e+02$	+	$\mathbf{8.92e+02 \pm 2.60e+02}$	=	$2.21e+03 \pm 1.99e+02$	+
f_{23}	$3.41e+02 \pm 6.99e-01$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.62e+02 \pm 6.87e+00$	+	$4.50e+02 \pm 1.86e+01$	+
f_{24}	$2.76e+02 \pm 4.60e+00$	$2.00e+\mathbf{02 \pm 0.00e+00}$	-	$3.14e+02 \pm 1.80e+01$	+	$3.66e+02 \pm 3.50e+01$	+
f_{25}	$2.15e+02 \pm 5.63e+00$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$2.34e+02 \pm 1.02e+01$	+	$2.74e+02 \pm 1.54e+01$	+
f_{26}	$1.24e+02 \pm 4.23e+01$	$2.00e+02 \pm 0.00e+00$	+	$1.32e+02 \pm 4.77e+01$	+	$\mathbf{1.18e+02 \pm 3.58e+01}$	-
f_{27}	$1.19e+03 \pm 7.85e+01$	$\mathbf{2.00e+02 \pm 1.36e-12}$	-	$1.33e+03 \pm 9.24e+01$	+	$2.11e+03 \pm 6.44e+01$	+
f_{28}	$2.03e+03 \pm 5.88e+02$	$2.00e+\mathbf{02 \pm 1.36e-12}$	-	$1.73e+03 \pm 2.49e+02$	=	$7.59e+03 \pm 7.25e+02$	+
f_{29}	$1.67e+03 \pm 5.49e+02$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$6.59e+03 \pm 2.28e+03$	+	$3.45e+08 \pm 9.91e+07$	+
f_{30}	$1.16e+04 \pm 1.38e+03$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$4.19e+04 \pm 1.66e+04$	+	$1.33e+06 \pm 6.71e+05$	+

Table B.17: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against the other compact algorithms (with RI) on CEC2014 in 100D.

	RIcDE	RIcBFO		RIcGA		RIcPSO	
f_1	$8.59e+07 \pm 2.56e+07$	$\mathbf{6.81e+07 \pm 6.95e+06}$	-	$5.11e+08 \pm 1.31e+08$	+	$2.29e+09 \pm 3.75e+08$	+
f_2	$4.40e+07 \pm 3.21e+07$	$\mathbf{8.72e+06 \pm 9.86e+05}$	-	$4.57e+10 \pm 1.15e+10$	+	$1.87e+11 \pm 1.29e+10$	+
f_3	$\mathbf{3.68e+04 \pm 1.36e+04}$	$1.07e+05 \pm 1.25e+04$	+	$4.19e+05 \pm 4.22e+04$	+	$3.76e+05 \pm 3.73e+04$	+
f_4	$\mathbf{4.44e+02 \pm 6.92e+01}$	$4.63e+02 \pm 4.63e+01$	=	$4.95e+03 \pm 1.05e+03$	+	$3.68e+04 \pm 4.43e+03$	+
f_5	$\mathbf{2.00e+01 \pm 2.54e-02}$	$2.09e+01 \pm 4.10e-02$	+	$2.06e+01 \pm 8.27e-02$	+	$2.13e+01 \pm 2.27e-02$	+
f_6	$\mathbf{9.57e+01 \pm 5.43e+00}$	$1.42e+02 \pm 3.17e+00$	+	$1.07e+02 \pm 6.45e+00$	+	$1.47e+02 \pm 3.31e+00$	+
f_7	$\mathbf{1.34e+00 \pm 5.34e-01}$	$1.42e+00 \pm 6.12e-02$	+	$3.62e+02 \pm 7.12e+01$	+	$1.72e+03 \pm 1.37e+02$	+
f_8	$\mathbf{2.03e+02 \pm 2.51e+01}$	$5.92e+02 \pm 1.17e+01$	+	$8.55e+02 \pm 9.64e+01$	+	$1.26e+03 \pm 3.26e+01$	+
f_9	$\mathbf{6.04e+02 \pm 6.31e+01}$	$7.73e+02 \pm 1.84e+01$	+	$1.03e+03 \pm 1.17e+02$	+	$1.42e+03 \pm 5.17e+01$	+
f_{10}	$\mathbf{4.83e+03 \pm 9.31e+02}$	$1.63e+04 \pm 3.57e+02$	+	$1.82e+04 \pm 1.15e+03$	+	$2.37e+04 \pm 1.65e+03$	+
f_{11}	$\mathbf{1.33e+04 \pm 8.56e+02}$	$1.34e+04 \pm 4.35e+02$	=	$1.89e+04 \pm 1.12e+03$	+	$3.06e+04 \pm 3.93e+02$	+
f_{12}	$\mathbf{5.29e-01 \pm 8.81e-02}$	$1.61e+00 \pm 3.02e-01$	+	$8.52e-01 \pm 1.93e-01$	+	$4.18e+00 \pm 2.12e-01$	+
f_{13}	$6.23e-01 \pm 6.06e-02$	$\mathbf{4.94e-01 \pm 4.01e-02}$	+	$2.18e+00 \pm 1.14e+00$	+	$7.08e+00 \pm 2.47e-01$	+
f_{14}	$4.00e-01 \pm 1.56e-01$	$\mathbf{2.97e-01 \pm 1.81e-02}$	-	$1.04e+02 \pm 2.06e+01$	+	$5.11e+02 \pm 4.19e+01$	+
f_{15}	$4.35e+02 \pm 2.21e+02$	$\mathbf{1.59e+02 \pm 1.72e+01}$	-	$5.28e+05 \pm 3.25e+05$	+	$4.93e+06 \pm 1.24e+06$	+
f_{16}	$\mathbf{4.38e+01 \pm 9.04e-01}$	$4.61e+01 \pm 2.10e-01$	+	$4.53e+01 \pm 5.39e-01$	+	$4.66e+01 \pm 2.24e-01$	+
f_{17}	$9.59e+06 \pm 4.52e+06$	$\mathbf{2.03e+06 \pm 3.34e+05}$	-	$5.72e+07 \pm 1.62e+07$	+	$2.29e+08 \pm 8.29e+07$	+
f_{18}	$\mathbf{3.45e+03 \pm 4.04e+03}$	$1.14e+05 \pm 1.89e+04$	+	$3.97e+05 \pm 8.70e+05$	+	$3.28e+09 \pm 7.15e+08$	+
f_{19}	$\mathbf{1.20e+02 \pm 1.69e+01}$	$1.47e+02 \pm 9.90e+00$	+	$2.32e+02 \pm 3.75e+01$	+	$1.27e+03 \pm 1.52e+02$	+
f_{20}	$9.02e+04 \pm 2.84e+04$	$\mathbf{6.78e+03 \pm 4.37e+03}$	-	$2.55e+05 \pm 6.41e+04$	+	$2.67e+05 \pm 9.35e+04$	+
f_{21}	$4.70e+06 \pm 1.79e+06$	$\mathbf{2.57e+06 \pm 4.91e+05}$	-	$2.70e+07 \pm 1.06e+07$	+	$7.30e+07 \pm 2.43e+07$	+
f_{22}	$\mathbf{2.24e+03 \pm 3.44e+02}$	$2.68e+03 \pm 3.85e+02$	+	$2.39e+03 \pm 4.31e+02$	=	$5.71e+03 \pm 3.48e+02$	+
f_{23}	$3.42e+02 \pm 2.73e+00$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$5.19e+02 \pm 4.55e+01$	+	$7.76e+02 \pm 1.19e+02$	+
f_{24}	$4.08e+02 \pm 2.66e+01$	$2.00e+\mathbf{02 \pm 0.00e+00}$	-	$6.08e+02 \pm 1.36e+02$	+	$6.85e+02 \pm 1.23e+02$	+
f_{25}	$2.67e+02 \pm 1.89e+01$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$3.31e+02 \pm 4.97e+01$	+	$4.58e+02 \pm 1.09e+02$	+
f_{26}	$2.05e+02 \pm 2.09e+00$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$2.29e+02 \pm 2.78e+01$	+	$3.04e+02 \pm 8.61e+01$	+
f_{27}	$2.76e+03 \pm 1.30e+02$	$\mathbf{2.00e+02 \pm 2.27e-12}$	-	$3.03e+03 \pm 1.22e+02$	+	$4.55e+03 \pm 1.25e+02$	+
f_{28}	$6.15e+03 \pm 1.24e+03$	$2.00e+\mathbf{02 \pm 2.27e-12}$	-	$5.62e+03 \pm 6.80e+02$	=	$2.08e+04 \pm 3.62e+03$	+
f_{29}	$4.99e+03 \pm 1.08e+03$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$5.80e+05 \pm 6.72e+05$	+	$1.63e+09 \pm 3.02e+08$	+
f_{30}	$5.30e+04 \pm 2.34e+04$	$\mathbf{2.00e+02 \pm 0.00e+00}$	-	$1.40e+06 \pm 5.13e+05$	+	$1.17e+07 \pm 4.56e+06$	+

Table B.18: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) for RIcDE against the other compact algorithms (with RI) on CEC2013-LSGO in 1000D.

	RIcDE	RIcBFO		RIcGA		RIcPSO	
f_1	$1.75e+11 \pm 1.13e+09$	$1.39e+11 \pm 1.45e+09$	-	$1.72e+11 \pm 1.26e+09$	-	$\mathbf{1.15e+11 \pm 1.73e+09}$	-
f_2	$1.88e+11 \pm 1.37e+08$	$2.06e+11 \pm 3.80e+08$	+	$\mathbf{1.88e+11 \pm 1.36e+08}$	-	$1.94e+11 \pm 4.19e+08$	+
f_3	$1.09e+11 \pm 3.42e+08$	$1.74e+11 \pm 1.24e+09$	+	$\mathbf{1.06e+11 \pm 3.61e+08}$	-	$1.53e+11 \pm 3.54e+09$	+
f_4	$1.74e+11 \pm 1.01e+09$	$1.34e+11 \pm 6.78e+08$	-	$1.91e+11 \pm 2.70e+09$	+	$\mathbf{1.17e+11 \pm 1.60e+09}$	-
f_5	$\mathbf{1.85e+11 \pm 1.71e+08}$	$2.06e+11 \pm 2.63e+08$	+	$1.90e+11 \pm 2.02e+08$	+	$1.95e+11 \pm 2.00e+08$	+
f_6	$1.21e+11 \pm 6.63e+08$	$1.99e+11 \pm 8.67e+08$	+	$9.66e+\mathbf{10 \pm 5.91e+08}$	-	$1.52e+11 \pm 1.84e+09$	+
f_7	$1.63e+11 \pm 6.64e+08$	$1.46e+11 \pm 1.04e+09$	-	$1.74e+11 \pm 1.88e+09$	+	$\mathbf{1.14e+11 \pm 2.78e+09}$	-
f_8	$2.10e+11 \pm 1.92e+09$	$1.49e+11 \pm 1.11e+09$	-	$1.61e+11 \pm 2.78e+09$	-	$\mathbf{1.22e+11 \pm 9.53e+08}$	-
f_9	$1.64e+11 \pm 3.54e+09$	$1.53e+11 \pm 1.05e+09$	-	$1.86e+11 \pm 2.99e+09$	+	$\mathbf{1.23e+11 \pm 6.94e+08}$	-
f_{10}	$1.35e+11 \pm 7.85e+08$	$1.95e+11 \pm 6.92e+08$	+	$\mathbf{9.69e+10 \pm 1.92e+08}$	-	$1.54e+11 \pm 3.28e+09$	+
f_{11}	$1.98e+11 \pm 2.73e+09$	$1.33e+11 \pm 4.99e+08$	-	$1.59e+11 \pm 1.87e+09$	-	$\mathbf{1.18e+11 \pm 7.85e+08}$	-
f_{12}	$2.06e+11 \pm 1.34e+09$	$\mathbf{1.09e+11 \pm 5.35e+08}$	-	$1.81e+11 \pm 5.93e+09$	-	$1.21e+11 \pm 1.33e+09$	-
f_{13}	$2.01e+11 \pm 2.33e+09$	$1.90e+11 \pm 1.10e+09$	-	$1.99e+11 \pm 2.15e+09$	-	$\mathbf{1.21e+11 \pm 1.03e+09}$	-
f_{14}	$1.83e+11 \pm 1.69e+09$	$1.95e+11 \pm 6.48e+08$	+	$1.93e+11 \pm 1.44e+09$	+	$\mathbf{1.14e+11 \pm 1.72e+09}$	-
f_{15}	$2.10e+11 \pm 1.68e+09$	$1.57e+11 \pm 1.18e+09$	-	$1.80e+11 \pm 1.82e+09$	-	$\mathbf{1.28e+11 \pm 1.74e+09}$	-

Appendix C. Detailed results on CEC2014 - Comparison against state-of-the-art single-solution algorithms

We report the detailed results of the RIcDE algorithm on the CEC2014 benchmark [43] in 10, 50 and 100 dimensions, respectively in Table C.19, C.20 and C.21, in comparison with: Non-Uniform Simulated Annealing (nuSA) [52], Covariance Matrix Adaptation Evolution Strategy 1+1, (CMA-ES 1+1) [41], and Parallel Memetic Structures (PMS). The parametrization of the algorithms is indicated in the main text.

Table C.19: Comparison against single-solution algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 10D.

	RIcDE	nuSA		CMA-ES 1+1		PMS	
f_1	$1.38e+05 \pm 1.55e+05$	$1.26e+05 \pm 4.78e+04$	=	$\mathbf{0.00e+00 \pm 0.00e+00}$	-	$1.20e+05 \pm 1.46e-11$	=
f_2	$8.96e+02 \pm 1.22e+03$	$1.38e+03 \pm 1.63e+03$	=	$\mathbf{0.00e+00 \pm 0.00e+00}$	-	$1.50e+03 \pm 6.82e-13$	+
f_3	$6.92e+02 \pm 6.26e+02$	$8.24e+02 \pm 5.23e+02$	=	$\mathbf{0.00e+00 \pm 0.00e+00}$	-	$2.42e+03 \pm 0.00e+00$	+
f_4	$\mathbf{2.76e+00 \pm 8.60e+00}$	$3.09e+01 \pm 1.21e+01$	+	$2.49e+01 \pm 1.51e+01$	+	$3.52e+01 \pm 1.71e-13$	+
f_5	$\mathbf{2.00e+01 \pm 3.23e-03}$	$2.00e+01 \pm 2.61e-02$	+	$2.00e+01 \pm 2.87e-03$	+	$2.00e+01 \pm 3.41e-13$	+
f_6	$\mathbf{1.16e+00 \pm 7.62e-01}$	$2.18e+00 \pm 1.67e+00$	+	$1.55e+01 \pm 2.29e+00$	+	$5.65e+00 \pm 1.14e-13$	+
f_7	$\mathbf{9.91e-02 \pm 4.62e-02}$	$2.58e-01 \pm 1.28e-01$	+	$1.52e-01 \pm 1.29e-01$	=	$2.83e-01 \pm 1.14e-13$	+
f_8	$6.63e-02 \pm 2.48e-01$	$1.34e+01 \pm 5.12e+00$	+	$1.36e+02 \pm 4.46e+01$	+	$\mathbf{1.73e-02 \pm 3.41e-13}$	-
f_9	$\mathbf{9.79e+00 \pm 3.44e+00}$	$1.31e+01 \pm 4.42e+00$	+	$1.54e+02 \pm 5.14e+01$	+	$1.02e+01 \pm 5.68e-13$	=
f_{10}	$3.84e+00 \pm 4.04e+00$	$3.21e+02 \pm 1.98e+02$	+	$1.66e+03 \pm 3.20e+02$	+	$\mathbf{3.67e-01 \pm 6.82e-13}$	-
f_{11}	$\mathbf{3.44e+02 \pm 1.62e+02}$	$6.08e+02 \pm 3.04e+02$	+	$1.77e+03 \pm 3.88e+02$	+	$8.74e+02 \pm 1.14e-12$	+
f_{12}	$\mathbf{6.47e-02 \pm 3.39e-02}$	$1.72e-01 \pm 1.24e-01$	+	$1.19e+00 \pm 1.19e+00$	+	$1.04e-01 \pm 4.55e-13$	+
f_{13}	$2.35e-01 \pm 8.82e-02$	$\mathbf{1.95e-01 \pm 9.14e-02}$	=	$5.34e-01 \pm 1.85e-01$	+	$4.69e-01 \pm 0.00e+00$	+
f_{14}	$2.41e-01 \pm 1.15e-01$	$\mathbf{1.66e-01 \pm 6.05e-02}$	-	$4.66e-01 \pm 3.08e-01$	+	$4.39e-01 \pm 2.27e-13$	+
f_{15}	$\mathbf{9.56e-01 \pm 3.66e-01}$	$1.25e+00 \pm 4.52e-01$	+	$3.53e+00 \pm 2.31e+00$	+	$2.51e+00 \pm 4.55e-13$	+
f_{16}	$\mathbf{2.16e+00 \pm 4.30e-01}$	$2.61e+00 \pm 4.81e-01$	+	$4.63e+00 \pm 3.04e-01$	+	$2.41e+00 \pm 1.14e-12$	+
f_{17}	$1.23e+04 \pm 1.73e+04$	$6.51e+03 \pm 4.73e+03$	=	$\mathbf{5.43e+02 \pm 2.98e+02}$	-	$5.78e+04 \pm 2.91e-11$	+
f_{18}	$2.96e+03 \pm 3.90e+03$	$1.27e+04 \pm 1.28e+04$	+	$\mathbf{3.88e+01 \pm 2.47e+01}$	-	$3.38e+04 \pm 7.28e-12$	+
f_{19}	$\mathbf{5.81e-01 \pm 4.08e-01}$	$1.99e+00 \pm 5.92e-01$	+	$7.40e+00 \pm 2.60e+00$	+	$1.56e+00 \pm 6.82e-13$	+
f_{20}	$8.91e+02 \pm 9.92e+02$	$7.39e+02 \pm 2.05e+03$	=	$\mathbf{1.16e+02 \pm 7.45e+01}$	-	$4.38e+03 \pm 3.64e-12$	+
f_{21}	$7.80e+02 \pm 1.08e+03$	$1.25e+03 \pm 1.02e+03$	+	$\mathbf{3.59e+02 \pm 2.10e+02}$	=	$5.12e+03 \pm 4.55e-12$	+
f_{22}	$\mathbf{4.64e+00 \pm 7.58e+00}$	$3.68e+01 \pm 2.74e+01$	+	$2.65e+02 \pm 1.48e+02$	+	$2.04e+01 \pm 1.36e-12$	+
f_{23}	$3.27e+02 \pm 9.21e+00$	$3.29e+02 \pm 2.38e-10$	+	$\mathbf{3.18e+02 \pm 5.91e+01}$	-	$3.30e+02 \pm 1.36e-12$	+
f_{24}	$\mathbf{1.23e+02 \pm 6.41e+00}$	$1.26e+02 \pm 1.01e+01$	=	$3.26e+02 \pm 1.78e+02$	+	$1.26e+02 \pm 4.55e-13$	+
f_{25}	$\mathbf{1.64e+02 \pm 3.03e+01}$	$1.94e+02 \pm 2.49e+01$	+	$1.99e+02 \pm 1.14e+01$	+	$1.65e+02 \pm 4.55e-13$	=
f_{26}	$1.00e+02 \pm 5.99e-02$	$\mathbf{1.00e+02 \pm 6.95e-02}$	-	$1.56e+02 \pm 9.08e+01$	+	$1.00e+02 \pm 1.36e-12$	+
f_{27}	$1.37e+02 \pm 1.73e+02$	$1.58e+02 \pm 1.90e+02$	=	$5.01e+02 \pm 1.43e+02$	+	$\mathbf{7.74e+00 \pm 0.00e+00}$	=
f_{28}	$4.19e+02 \pm 4.55e+01$	$4.09e+02 \pm 5.57e+01$	=	$2.83e+03 \pm 1.65e+03$	+	$\mathbf{4.01e+02 \pm 0.00e+00}$	=
f_{29}	$3.14e+02 \pm 5.70e+01$	$5.86e+04 \pm 3.10e+05$	+	$1.19e+05 \pm 4.43e+05$	=	$\mathbf{2.65e+02 \pm 1.36e-12}$	-
f_{30}	$6.60e+\mathbf{02 \pm 1.55e+02}$	$1.12e+03 \pm 3.57e+02$	+	$1.38e+03 \pm 4.85e+02$	+	$1.39e+03 \pm 1.82e-12$	+

Table C.20: Comparison against single-solution algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 50D.

	RIcDE	nuSA		CMA-ES 1+1		PMS	
f_1	$7.12e+06 \pm 2.83e+06$	$5.67e+06 \pm 6.24e+05$	=	$\mathbf{2.71e+03 \pm 1.71e+03}$	-	$4.74e+07 \pm 7.45e-09$	+
f_2	$1.67e+03 \pm 3.41e+03$	$9.99e+07 \pm 2.79e+07$	+	$\mathbf{2.84e-14 \pm 2.65e-14}$	-	$3.80e+07 \pm 2.24e-08$	+
f_3	$1.86e+04 \pm 7.37e+03$	$1.40e+04 \pm 1.57e+03$	-	$\mathbf{5.68e-14 \pm 4.40e-14}$	-	$2.56e+04 \pm 0.00e+00$	+
f_4	$1.24e+02 \pm 2.19e+01$	$2.08e+02 \pm 4.36e+01$	+	$\mathbf{5.58e+01 \pm 4.82e+01}$	-	$2.55e+02 \pm 2.27e-13$	+
f_5	$2.00e+01 \pm 9.92e-06$	$2.07e+01 \pm 1.44e-01$	+	$\mathbf{2.00e+01 \pm 2.11e-06}$	-	$2.00e+01 \pm 0.00e+00$	+
f_6	$\mathbf{3.25e+01 \pm 2.81e+00}$	$3.63e+01 \pm 5.89e+00$	+	$8.38e+01 \pm 5.45e+00$	+	$4.15e+01 \pm 1.14e-13$	+
f_7	$2.35e-02 \pm 1.61e-02$	$3.85e+00 \pm 4.05e-01$	+	$\mathbf{6.31e-03 \pm 1.03e-02}$	-	$1.71e+00 \pm 0.00e+00$	+
f_8	$4.09e+01 \pm 7.74e+00$	$1.63e+02 \pm 2.27e+01$	+	$6.92e+02 \pm 9.80e+01$	+	$\mathbf{1.48e+01 \pm 5.68e-13}$	-
f_9	$1.90e+02 \pm 2.94e+01$	$\mathbf{1.72e+02 \pm 2.67e+01}$	-	$1.06e+03 \pm 2.29e+02$	+	$2.01e+02 \pm 0.00e+00$	+
f_{10}	$9.08e+02 \pm 3.20e+02$	$5.51e+03 \pm 8.68e+02$	+	$8.20e+03 \pm 1.19e+03$	+	$\mathbf{9.77e+00 \pm 0.00e+00}$	-
f_{11}	$\mathbf{4.92e+03 \pm 6.44e+02}$	$6.93e+03 \pm 1.30e+03$	+	$8.31e+03 \pm 7.27e+02$	+	$6.39e+03 \pm 3.64e-12$	+
f_{12}	$\mathbf{2.05e-01 \pm 5.80e-02}$	$7.81e-01 \pm 4.42e-01$	+	$1.73e+00 \pm 8.17e-01$	+	$4.34e-01 \pm 4.55e-13$	+
f_{13}	$5.46e-01 \pm 7.83e-02$	$6.44e-01 \pm 1.06e-01$	+	$5.99e-01 \pm 1.11e-01$	=	$\mathbf{4.65e-01 \pm 2.27e-13}$	-
f_{14}	$3.38e-01 \pm 1.46e-01$	$\mathbf{2.36e-01 \pm 2.95e-02}$	-	$4.10e-01 \pm 2.05e-01$	+	$2.46e-01 \pm 2.27e-13$	-
f_{15}	$\mathbf{2.81e+01 \pm 7.03e+00}$	$3.32e+01 \pm 5.32e+00$	+	$6.21e+01 \pm 1.77e+01$	+	$6.64e+01 \pm 6.82e-13$	+
f_{16}	$\mathbf{2.01e+01 \pm 3.84e-01}$	$2.07e+01 \pm 7.32e-01$	+	$2.37e+01 \pm 4.72e-01$	+	$2.01e+01 \pm 2.27e-13$	=
f_{17}	$1.21e+06 \pm 6.41e+05$	$2.26e+05 \pm 7.12e+04$	-	$\mathbf{2.78e+03 \pm 4.85e+02}$	-	$1.63e+07 \pm 1.30e-08$	+
f_{18}	$9.33e+02 \pm 1.24e+03$	$1.57e+03 \pm 1.23e+03$	+	$5.63e+02 \pm 1.02e+02$	=	$\mathbf{1.66e+02 \pm 2.27e-13}$	-
f_{19}	$\mathbf{3.43e+01 \pm 1.68e+01}$	$3.78e+01 \pm 2.35e+01$	=	$7.22e+01 \pm 2.41e+01$	+	$8.60e+01 \pm 6.82e-13$	+
f_{20}	$2.95e+04 \pm 1.19e+04$	$9.99e+02 \pm 2.81e+02$	-	$\mathbf{5.29e+02 \pm 1.03e+02}$	-	$3.15e+04 \pm 1.46e-11$	+
f_{21}	$6.52e+05 \pm 5.03e+05$	$1.48e+05 \pm 4.94e+04$	-	$\mathbf{2.21e+03 \pm 3.98e+02}$	-	$9.68e+06 \pm 3.73e-09$	+
f_{22}	$9.37e+02 \pm 2.52e+02$	$\mathbf{8.45e+02 \pm 2.80e+02}$	=	$1.58e+03 \pm 3.99e+02$	+	$1.81e+03 \pm 1.82e-12$	+
f_{23}	$3.41e+02 \pm 6.99e-01$	$3.57e+02 \pm 7.19e-01$	+	$\mathbf{3.41e+02 \pm 1.30e-04}$	=	$3.97e+02 \pm 0.00e+00$	+
f_{24}	$2.76e+02 \pm 4.60e+00$	$\mathbf{2.60e+02 \pm 2.88e+00}$	-	$7.26e+02 \pm 3.89e+02$	+	$2.78e+02 \pm 0.00e+00$	+
f_{25}	$\mathbf{2.15e+02 \pm 5.63e+00}$	$2.24e+02 \pm 1.95e+00$	+	$2.52e+02 \pm 2.11e+01$	+	$2.30e+02 \pm 4.55e-13$	+
f_{26}	$1.24e+02 \pm 4.23e+01$	$1.11e+02 \pm 2.98e+01$	-	$1.75e+02 \pm 1.11e+02$	=	$\mathbf{1.01e+02 \pm 0.00e+00}$	-
f_{27}	$\mathbf{1.19e+03 \pm 7.85e+01}$	$1.28e+03 \pm 1.21e+02$	+	$2.34e+03 \pm 4.32e+02$	+	$1.39e+03 \pm 2.73e-12$	+
f_{28}	$2.03e+03 \pm 5.88e+02$	$3.99e+03 \pm 8.16e+02$	+	$1.46e+04 \pm 3.46e+03$	+	$\mathbf{1.78e+03 \pm 1.82e-12}$	=
f_{29}	$\mathbf{1.67e+03 \pm 5.49e+02}$	$2.39e+07 \pm 7.18e+07$	+	$6.68e+06 \pm 2.51e+07$	+	$2.34e+03 \pm 9.09e-13$	+
f_{30}	$\mathbf{1.16e+04 \pm 1.38e+03}$	$7.46e+04 \pm 7.82e+03$	+	$2.66e+04 \pm 9.73e+03$	+	$1.54e+04 \pm 3.64e-12$	+

Table C.21: Comparison against single-solution algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 100D.

	RIcDE	nuSA		CMA-ES 1+1		PMS	
f_1	$8.59e+07 \pm 2.56e+07$	$4.52e+07 \pm 1.66e+06$	-	$\mathbf{1.02e+05 \pm 3.65e+04}$	-	$3.22e+08 \pm 2.38e-07$	+
f_2	$4.40e+07 \pm 3.21e+07$	$8.07e+09 \pm 5.71e+08$	+	$\mathbf{2.51e+02 \pm 2.58e+02}$	-	$6.91e+08 \pm 3.58e-07$	+
f_3	$3.68e+04 \pm 1.36e+04$	$5.41e+04 \pm 2.56e+03$	+	$\mathbf{5.68e-14 \pm 1.45e-13}$	-	$5.78e+04 \pm 1.46e-11$	+
f_4	$4.44e+02 \pm 6.92e+01$	$6.97e+02 \pm 8.37e+01$	+	$\mathbf{2.01e+02 \pm 6.39e+01}$	-	$6.56e+02 \pm 0.00e+00$	+
f_5	$2.00e+01 \pm 2.54e-02$	$2.09e+01 \pm 1.03e-01$	+	$\mathbf{2.00e+01 \pm 7.43e-07}$	-	$2.00e+01 \pm 2.27e-13$	-
f_6	$\mathbf{9.57e+01 \pm 5.43e+00}$	$9.75e+01 \pm 5.32e+00$	=	$1.69e+02 \pm 7.72e+00$	+	$1.08e+02 \pm 2.27e-13$	+
f_7	$1.34e+00 \pm 5.34e-01$	$1.05e+02 \pm 7.28e+00$	+	$\mathbf{4.02e-03 \pm 6.08e-03}$	-	$7.68e+00 \pm 0.00e+00$	+
f_8	$2.03e+02 \pm 2.51e+01$	$4.09e+02 \pm 4.22e+01$	+	$1.29e+03 \pm 1.84e+02$	+	$\mathbf{4.82e+01 \pm 0.00e+00}$	-
f_9	$6.04e+02 \pm 6.31e+01$	$\mathbf{4.50e+02 \pm 3.44e+01}$	-	$1.94e+03 \pm 2.46e+02$	+	$7.03e+02 \pm 1.14e-12$	+
f_{10}	$4.83e+03 \pm 9.31e+02$	$1.35e+04 \pm 2.03e+03$	+	$1.65e+04 \pm 1.21e+03$	+	$\mathbf{1.59e+02 \pm 0.00e+00}$	-
f_{11}	$\mathbf{1.33e+04 \pm 8.56e+02}$	$1.85e+04 \pm 3.53e+03$	+	$1.70e+04 \pm 1.65e+03$	+	$1.37e+04 \pm 9.09e-12$	=
f_{12}	$\mathbf{5.29e-01 \pm 8.81e-02}$	$1.42e+00 \pm 4.40e-01$	+	$1.92e+00 \pm 5.62e-01$	+	$5.64e-01 \pm 6.82e-13$	+
f_{13}	$6.23e-01 \pm 6.06e-02$	$6.27e-01 \pm 5.38e-02$	=	$\mathbf{6.12e-01 \pm 8.09e-02}$	=	$6.51e-01 \pm 4.55e-13$	+
f_{14}	$4.00e-01 \pm 1.56e-01$	$\mathbf{2.01e-01 \pm 7.18e-03}$	-	$3.55e-01 \pm 9.32e-02$	=	$7.36e-01 \pm 2.27e-13$	+
f_{15}	$4.35e+02 \pm 2.21e+02$	$3.22e+02 \pm 4.96e+01$	-	$\mathbf{1.95e+02 \pm 6.46e+01}$	-	$3.39e+02 \pm 2.27e-13$	=
f_{16}	$4.38e+01 \pm 9.04e-01$	$4.33e+01 \pm 1.07e+00$	-	$4.75e+01 \pm 7.14e-01$	+	$\mathbf{4.24e+01 \pm 1.14e-12}$	-
f_{17}	$9.59e+06 \pm 4.52e+06$	$1.34e+06 \pm 1.74e+05$	-	$\mathbf{7.58e+03 \pm 1.35e+03}$	-	$8.74e+07 \pm 4.47e-08$	+
f_{18}	$3.45e+03 \pm 4.04e+03$	$\mathbf{1.24e+03 \pm 3.75e+02}$	=	$1.37e+03 \pm 2.07e+02$	=	$3.17e+07 \pm 1.49e-08$	+
f_{19}	$1.20e+02 \pm 1.69e+01$	$\mathbf{1.12e+02 \pm 3.15e+01}$	=	$1.80e+02 \pm 4.49e+01$	+	$1.09e+03 \pm 4.55e-13$	+
f_{20}	$9.02e+04 \pm 2.84e+04$	$8.46e+03 \pm 1.48e+03$	-	$\mathbf{1.20e+03 \pm 2.36e+02}$	-	$8.08e+04 \pm 1.46e-11$	-
f_{21}	$4.70e+06 \pm 1.79e+06$	$8.27e+05 \pm 1.29e+05$	-	$\mathbf{6.35e+03 \pm 8.36e+02}$	-	$2.38e+07 \pm 3.73e-09$	+
f_{22}	$2.24e+03 \pm 3.44e+02$	$\mathbf{2.08e+03 \pm 5.25e+02}$	=	$3.02e+03 \pm 5.23e+02$	+	$2.69e+03 \pm 3.64e-12$	+
f_{23}	$3.42e+02 \pm 2.73e+00$	$4.00e+02 \pm 1.43e+00$	+	$\mathbf{3.40e+02 \pm 3.87e-02}$	-	$1.93e+03 \pm 1.82e-12$	+
f_{24}	$4.08e+02 \pm 2.66e+01$	$3.38e+02 \pm 3.21e+01$	-	$1.32e+03 \pm 7.50e+02$	+	$3.92e+02 \pm 9.09e-13$	-
f_{25}	$2.67e+02 \pm 1.89e+01$	$\mathbf{2.00e+02 \pm 3.07e-10}$	-	$3.33e+02 \pm 3.48e+01$	+	$2.74e+02 \pm 0.00e+00$	=
f_{26}	$2.05e+02 \pm 2.09e+00$	$\mathbf{2.00e+02 \pm 1.33e+02}$	-	$2.00e+02 \pm 1.32e-01$	-	$2.20e+02 \pm 4.55e-13$	+
f_{27}	$2.76e+03 \pm 1.30e+02$	$2.84e+03 \pm 2.07e+02$	=	$5.17e+03 \pm 1.01e+03$	+	$\mathbf{2.48e+03 \pm 2.73e-12}$	-
f_{28}	$\mathbf{6.15e+03 \pm 1.24e+03}$	$1.50e+04 \pm 1.09e+03$	+	$3.32e+04 \pm 5.97e+03$	+	$6.56e+03 \pm 5.46e-12$	+
f_{29}	$\mathbf{4.99e+03 \pm 1.08e+03}$	$6.79e+03 \pm 1.22e+03$	+	$9.57e+03 \pm 3.65e+03$	+	$7.28e+07 \pm 4.47e-08$	+
f_{30}	$5.30e+04 \pm 2.34e+04$	$1.60e+05 \pm 1.19e+04$	+	$\mathbf{2.53e+04 \pm 6.24e+03}$	-	$9.44e+06 \pm 5.59e-09$	+

Appendix D. Detailed results on CEC2014 - Comparison against state-of-the-art population-based algorithms

We report the detailed results of the RIcDE algorithm on the CEC2014 benchmark [43] in 10, 50 and 100 dimensions, respectively in Table D.22, D.23 and D.24, in comparison with: Adaptive Differential Evolution With Optional Archive (JADE) [54], Comprehensive Learning Particle Swarm Optimization (CLPSO) [55], and Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search (EPSDE-LS) [56]. The parametrization of the algorithms is indicated in the main text.

Table D.22: Comparison against population-based algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 10D.

	RIcDE	JADE		CLPSO		EPSDE-LS	
f_1	$1.38e+05 \pm 1.55e+05$	$1.22e+07 \pm 8.49e+06$	+	$8.27e+05 \pm 6.69e+05$	+	$\mathbf{1.04e+01 \pm 7.50e+00}$	-
f_2	$8.96e+02 \pm 1.22e+03$	$2.05e+08 \pm 1.89e+08$	+	$6.78e+02 \pm 4.35e+02$	=	$\mathbf{1.73e-06 \pm 1.86e-06}$	-
f_3	$6.92e+02 \pm 6.26e+02$	$1.83e+04 \pm 1.15e+04$	+	$1.95e+02 \pm 1.29e+02$	-	$\mathbf{6.42e-10 \pm 4.23e-10}$	-
f_4	$\mathbf{2.76e+00 \pm 8.60e+00}$	$8.02e+01 \pm 3.01e+01$	+	$7.48e+00 \pm 7.13e+00$	+	$1.12e+01 \pm 1.55e+01$	=
f_5	$2.00e+01 \pm 3.23e-03$	$2.04e+01 \pm 1.14e-01$	+	$1.98e+01 \pm 1.21e+00$	-	$\mathbf{1.96e+01 \pm 1.26e+00}$	-
f_6	$1.16e+00 \pm 7.62e-01$	$8.61e+00 \pm 7.34e-01$	+	$1.24e+00 \pm 3.73e-01$	=	$\mathbf{1.55e-01 \pm 2.55e-01}$	-
f_7	$9.91e-02 \pm 4.62e-02$	$4.24e+00 \pm 3.14e+00$	+	$1.28e-01 \pm 4.05e-02$	+	$\mathbf{5.22e-02 \pm 3.25e-02}$	-
f_8	$6.63e-02 \pm 2.48e-01$	$4.47e+01 \pm 8.17e+00$	+	$1.32e-04 \pm 1.01e-04$	-	$\mathbf{1.36e-12 \pm 7.07e-12}$	-
f_9	$9.79e+00 \pm 3.44e+00$	$5.13e+01 \pm 8.34e+00$	+	$1.04e+01 \pm 2.45e+00$	=	$\mathbf{7.58e+00 \pm 2.39e+00}$	-
f_{10}	$3.84e+00 \pm 4.04e+00$	$1.08e+03 \pm 2.45e+02$	+	$\mathbf{1.51e+00 \pm 1.05e+00}$	=	$1.14e+01 \pm 6.43e+00$	+
f_{11}	$\mathbf{3.44e+02 \pm 1.62e+02}$	$1.47e+03 \pm 2.21e+02$	+	$3.85e+02 \pm 1.10e+02$	=	$4.73e+02 \pm 1.97e+02$	+
f_{12}	$\mathbf{6.47e-02 \pm 3.39e-02}$	$1.64e+00 \pm 3.68e-01$	+	$4.04e-01 \pm 8.63e-02$	+	$2.23e-01 \pm 1.02e-01$	+
f_{13}	$2.35e-01 \pm 8.82e-02$	$5.85e-01 \pm 1.61e-01$	+	$1.79e-01 \pm 3.75e-02$	-	$\mathbf{1.06e-01 \pm 3.90e-02}$	-
f_{14}	$2.41e-01 \pm 1.15e-01$	$6.72e-01 \pm 5.00e-01$	+	$1.95e-01 \pm 3.73e-02$	=	$\mathbf{1.49e-01 \pm 3.98e-02}$	-
f_{15}	$9.56e-01 \pm 3.66e-01$	$9.88e+00 \pm 3.74e+00$	+	$1.54e+00 \pm 3.00e-01$	+	$\mathbf{7.93e-01 \pm 2.85e-01}$	=
f_{16}	$\mathbf{2.16e+00 \pm 4.30e-01}$	$3.74e+00 \pm 2.55e-01$	+	$2.30e+00 \pm 1.76e-01$	=	$2.37e+00 \pm 4.15e-01$	+
f_{17}	$1.23e+04 \pm 1.73e+04$	$2.51e+05 \pm 3.10e+05$	+	$9.01e+03 \pm 6.70e+03$	=	$\mathbf{5.26e+01 \pm 2.25e+01}$	-
f_{18}	$2.96e+03 \pm 3.90e+03$	$1.61e+05 \pm 2.96e+05$	+	$5.95e+01 \pm 4.73e+01$	-	$\mathbf{2.37e+00 \pm 1.15e+00}$	-
f_{19}	$5.81e-01 \pm 4.08e-01$	$5.01e+00 \pm 8.48e-01$	+	$7.48e-01 \pm 2.23e-01$	+	$\mathbf{4.71e-01 \pm 1.86e-01}$	=
f_{20}	$8.91e+02 \pm 9.92e+02$	$5.18e+03 \pm 5.73e+03$	+	$2.18e+01 \pm 1.66e+01$	-	$\mathbf{7.39e-01 \pm 3.48e-01}$	-
f_{21}	$7.80e+02 \pm 1.08e+03$	$2.53e+04 \pm 4.90e+04$	+	$8.83e+02 \pm 5.44e+02$	+	$\mathbf{5.13e+00 \pm 4.02e+00}$	-
f_{22}	$\mathbf{4.64e+00 \pm 7.58e+00}$	$1.01e+02 \pm 4.77e+01$	+	$1.14e+01 \pm 5.68e+00$	+	$5.10e+00 \pm 2.71e+00$	+
f_{23}	$3.27e+02 \pm 9.21e+00$	$3.37e+02 \pm 5.09e+00$	+	$\mathbf{3.00e+02 \pm 7.43e+01}$	-	$3.29e+02 \pm 9.09e-13$	+
f_{24}	$1.23e+02 \pm 6.41e+00$	$1.64e+02 \pm 9.65e+00$	+	$1.20e+02 \pm 3.84e+00$	-	$\mathbf{1.15e+02 \pm 4.01e+00}$	-
f_{25}	$1.64e+02 \pm 3.03e+01$	$1.98e+02 \pm 8.41e+00$	+	$1.57e+02 \pm 1.09e+01$	=	$\mathbf{1.33e+02 \pm 1.28e+01}$	-
f_{26}	$1.00e+02 \pm 5.99e-02$	$1.01e+02 \pm 1.60e-01$	+	$1.00e+02 \pm 3.70e-02$	=	$\mathbf{1.00e+02 \pm 3.37e-02}$	-
f_{27}	$1.37e+02 \pm 1.73e+02$	$2.68e+02 \pm 1.62e+02$	+	$\mathbf{7.54e+00 \pm 2.27e+00}$	=	$4.41e+01 \pm 1.06e+02$	-
f_{28}	$4.19e+02 \pm 4.55e+01$	$7.91e+02 \pm 1.03e+02$	+	$4.99e+02 \pm 6.74e+01$	+	$\mathbf{3.75e+02 \pm 2.34e+01}$	-
f_{29}	$3.14e+02 \pm 5.70e+01$	$1.54e+05 \pm 1.79e+05$	+	$3.19e+02 \pm 4.33e+01$	=	$\mathbf{2.23e+02 \pm 6.14e+00}$	-
f_{30}	$6.60e+02 \pm 1.55e+02$	$4.99e+03 \pm 2.68e+03$	+	$1.15e+03 \pm 2.06e+02$	+	$\mathbf{5.96e+02 \pm 3.59e+01}$	=

Table D.23: Comparison against population-based algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 50D.

	RIcDE	JADE		CLPSO		EPSDE-LS	
f_1	$7.12e+06 \pm 2.83e+06$	$1.75e+08 \pm 6.09e+07$	+	$2.62e+07 \pm 5.82e+06$	+	$\mathbf{1.46e+06 \pm 5.04e+05}$	-
f_2	$1.67e+03 \pm 3.41e+03$	$2.24e+09 \pm 4.79e+09$	+	$7.40e+04 \pm 3.68e+04$	+	$\mathbf{1.49e-02 \pm 2.24e-02}$	-
f_3	$1.86e+04 \pm 7.37e+03$	$8.85e+04 \pm 2.43e+04$	+	$6.23e+03 \pm 2.11e+03$	-	$\mathbf{2.04e-02 \pm 4.07e-02}$	-
f_4	$1.24e+02 \pm 2.19e+01$	$6.00e+02 \pm 4.79e+02$	+	$2.42e+02 \pm 2.00e+01$	+	$\mathbf{2.02e+01 \pm 2.73e+01}$	-
f_5	$\mathbf{2.00e+01 \pm 9.92e-06}$	$2.09e+01 \pm 5.19e-02$	+	$2.06e+01 \pm 4.33e-02$	+	$2.00e+01 \pm 8.07e-03$	=
f_6	$\mathbf{3.25e+01 \pm 2.81e+00}$	$5.65e+01 \pm 2.00e+00$	+	$3.29e+01 \pm 2.14e+00$	=	$3.73e+01 \pm 1.55e+00$	+
f_7	$2.35e-02 \pm 1.61e-02$	$3.25e+01 \pm 7.00e+01$	+	$1.25e-01 \pm 4.40e-02$	+	$\mathbf{1.72e-03 \pm 4.18e-03}$	-
f_8	$4.09e+01 \pm 7.74e+00$	$2.95e+02 \pm 4.87e+01$	+	$\mathbf{6.18e-02 \pm 1.76e-01}$	-	$1.06e-01 \pm 2.87e-01$	-
f_9	$1.90e+02 \pm 2.94e+01$	$3.75e+02 \pm 3.82e+01$	+	$1.86e+02 \pm 1.50e+01$	=	$\mathbf{1.33e+02 \pm 3.47e+01}$	-
f_{10}	$9.08e+02 \pm 3.20e+02$	$7.92e+03 \pm 6.17e+02$	+	$\mathbf{3.21e+01 \pm 7.77e+00}$	-	$3.71e+02 \pm 2.03e+02$	-
f_{11}	$\mathbf{4.92e+03 \pm 6.44e+02}$	$1.09e+04 \pm 6.37e+02$	+	$6.39e+03 \pm 4.75e+02$	+	$5.95e+03 \pm 6.81e+02$	+
f_{12}	$\mathbf{2.05e-01 \pm 5.80e-02}$	$1.83e+00 \pm 2.68e-01$	+	$6.02e-01 \pm 8.98e-02$	+	$5.20e-01 \pm 1.47e-01$	+
f_{13}	$5.46e-01 \pm 7.83e-02$	$1.66e+00 \pm 1.36e+00$	+	$4.18e-01 \pm 5.21e-02$	-	$\mathbf{3.86e-01 \pm 6.04e-02}$	-
f_{14}	$3.38e-01 \pm 1.46e-01$	$1.50e+01 \pm 2.42e+01$	+	$2.96e-01 \pm 2.79e-02$	=	$\mathbf{2.86e-01 \pm 2.84e-02}$	=
f_{15}	$2.81e+01 \pm 7.03e+00$	$1.46e+03 \pm 3.14e+03$	+	$2.66e+01 \pm 1.28e+00$	=	$\mathbf{1.29e+01 \pm 3.15e+00}$	-
f_{16}	$2.01e+01 \pm 3.84e-01$	$2.18e+01 \pm 4.72e-01$	+	$1.95e+01 \pm 4.09e-01$	-	$\mathbf{1.92e+01 \pm 6.00e-01}$	-
f_{17}	$1.21e+06 \pm 6.41e+05$	$5.08e+07 \pm 3.25e+07$	+	$4.62e+06 \pm 1.44e+06$	+	$\mathbf{4.35e+04 \pm 3.06e+04}$	-
f_{18}	$9.33e+02 \pm 1.24e+03$	$1.44e+07 \pm 3.70e+07$	+	$7.97e+02 \pm 2.87e+02$	-	$\mathbf{1.28e+02 \pm 6.16e+01}$	-
f_{19}	$3.43e+01 \pm 1.68e+01$	$1.69e+02 \pm 1.63e+02$	+	$2.34e+01 \pm 5.79e+00$	-	$\mathbf{1.19e+01 \pm 2.64e+00}$	-
f_{20}	$2.95e+04 \pm 1.19e+04$	$7.19e+04 \pm 5.04e+04$	+	$1.10e+04 \pm 3.16e+03$	-	$\mathbf{1.03e+02 \pm 5.18e+01}$	-
f_{21}	$6.52e+05 \pm 5.03e+05$	$1.36e+07 \pm 5.77e+06$	+	$1.86e+06 \pm 7.67e+05$	+	$\mathbf{1.27e+04 \pm 1.22e+04}$	-
f_{22}	$9.37e+02 \pm 2.52e+02$	$2.07e+03 \pm 2.26e+02$	+	$7.70e+02 \pm 1.37e+02$	-	$\mathbf{6.79e+02 \pm 1.97e+02}$	-
f_{23}	$3.41e+02 \pm 6.99e-01$	$3.50e+02 \pm 6.50e+00$	+	$3.43e+02 \pm 3.35e-01$	+	$\mathbf{3.41e+02 \pm 3.62e-13}$	-
f_{24}	$2.76e+02 \pm 4.60e+00$	$2.97e+02 \pm 1.24e+01$	+	$\mathbf{2.59e+02 \pm 3.02e+00}$	-	$2.69e+02 \pm 2.12e+00$	-
f_{25}	$2.15e+02 \pm 5.63e+00$	$2.63e+02 \pm 2.07e+01$	+	$2.28e+02 \pm 2.08e+00$	+	$\mathbf{2.11e+02 \pm 1.67e+00}$	-
f_{26}	$1.24e+02 \pm 4.23e+01$	$1.90e+02 \pm 3.37e+01$	+	$1.39e+02 \pm 4.77e+01$	+	$\mathbf{1.07e+02 \pm 2.49e+01}$	-
f_{27}	$1.19e+03 \pm 7.85e+01$	$1.61e+03 \pm 9.36e+01$	+	$\mathbf{9.83e+02 \pm 1.86e+02}$	-	$1.11e+03 \pm 7.62e+01$	-
f_{28}	$2.03e+03 \pm 5.88e+02$	$6.08e+03 \pm 8.79e+02$	+	$4.42e+03 \pm 5.93e+02$	+	$\mathbf{1.75e+03 \pm 2.23e+02}$	=
f_{29}	$1.67e+03 \pm 5.49e+02$	$1.71e+07 \pm 3.22e+07$	+	$2.25e+04 \pm 8.19e+03$	+	$\mathbf{1.31e+03 \pm 2.17e+02}$	-
f_{30}	$1.16e+04 \pm 1.38e+03$	$4.58e+05 \pm 4.46e+05$	+	$4.11e+04 \pm 6.41e+03$	+	$\mathbf{1.00e+04 \pm 6.66e+02}$	-

Table D.24: Comparison against population-based algorithms: Average error ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDE) on CEC2014 in 100D.

	RIcDE	JADE		CLPSO		EPSDE-LS	
f_1	$8.59e+07 \pm 2.56e+07$	$1.53e+08 \pm 1.15e+08$	+	$1.34e+08 \pm 2.39e+07$	+	$\mathbf{1.16e+07 \pm 7.02e+06}$	-
f_2	$4.40e+07 \pm 3.21e+07$	$6.21e+08 \pm 8.95e+08$	+	$3.70e+04 \pm 9.54e+03$	-	$\mathbf{4.96e+03 \pm 7.73e+03}$	-
f_3	$3.68e+04 \pm 1.36e+04$	$1.73e+05 \pm 6.41e+04$	+	$9.36e+03 \pm 2.16e+03$	-	$\mathbf{4.84e-01 \pm 5.91e-01}$	-
f_4	$4.44e+02 \pm 6.92e+01$	$1.88e+03 \pm 4.15e+03$	=	$4.38e+02 \pm 1.98e+01$	=	$\mathbf{7.26e+01 \pm 5.78e+01}$	-
f_5	$2.00e+01 \pm 2.54e-02$	$2.10e+01 \pm 8.39e-02$	+	$2.08e+01 \pm 3.68e-02$	+	$\mathbf{2.00e+01 \pm 6.47e-03}$	-
f_6	$9.57e+01 \pm 5.43e+00$	$1.24e+02 \pm 2.19e+00$	+	$\mathbf{9.20e+01 \pm 3.10e+00}$	-	$9.20e+01 \pm 2.53e+00$	-
f_7	$1.34e+00 \pm 5.34e-01$	$1.76e+01 \pm 2.72e+01$	+	$9.14e-02 \pm 1.86e-02$	-	$\mathbf{6.56e-04 \pm 3.53e-03}$	-
f_8	$2.03e+02 \pm 2.51e+01$	$6.49e+02 \pm 2.33e+02$	+	$\mathbf{2.47e-01 \pm 3.37e-01}$	-	$4.10e+00 \pm 1.59e+00$	-
f_9	$6.04e+02 \pm 6.31e+01$	$9.62e+02 \pm 2.35e+02$	+	$5.82e+02 \pm 3.29e+01$	=	$\mathbf{4.12e+02 \pm 8.89e+01}$	-
f_{10}	$4.83e+03 \pm 9.31e+02$	$1.84e+04 \pm 4.96e+03$	+	$\mathbf{1.38e+02 \pm 7.21e+01}$	-	$1.44e+03 \pm 4.49e+02$	-
f_{11}	$\mathbf{1.33e+04 \pm 8.56e+02}$	$2.41e+04 \pm 1.11e+03$	+	$1.58e+04 \pm 5.91e+02$	+	$1.50e+04 \pm 1.27e+03$	+
f_{12}	$\mathbf{5.29e-01 \pm 8.81e-02}$	$2.02e+00 \pm 1.89e-01$	+	$8.39e-01 \pm 8.36e-02$	+	$8.10e-01 \pm 1.94e-01$	+
f_{13}	$6.23e-01 \pm 6.06e-02$	$1.80e+00 \pm 2.08e+00$	=	$\mathbf{4.37e-01 \pm 2.91e-02}$	-	$4.48e-01 \pm 4.68e-02$	-
f_{14}	$4.00e-01 \pm 1.56e-01$	$1.98e+01 \pm 2.84e+01$	+	$3.13e-01 \pm 1.48e-02$	-	$\mathbf{2.89e-01 \pm 1.97e-02}$	-
f_{15}	$4.35e+02 \pm 2.21e+02$	$1.55e+05 \pm 3.94e+05$	=	$6.92e+01 \pm 3.84e+00$	-	$\mathbf{3.91e+01 \pm 7.54e+00}$	-
f_{16}	$4.38e+01 \pm 9.04e-01$	$4.50e+01 \pm 9.04e-01$	+	$4.23e+01 \pm 4.30e-01$	-	$\mathbf{4.17e+01 \pm 6.39e-01}$	-
f_{17}	$9.59e+06 \pm 4.52e+06$	$1.89e+08 \pm 1.67e+08$	+	$2.88e+07 \pm 6.16e+06$	+	$\mathbf{1.07e+06 \pm 4.40e+05}$	-
f_{18}	$3.45e+03 \pm 4.04e+03$	$2.14e+06 \pm 9.96e+06$	+	$1.74e+03 \pm 1.18e+03$	=	$\mathbf{5.97e+02 \pm 5.66e+02}$	-
f_{19}	$1.20e+02 \pm 1.69e+01$	$1.22e+02 \pm 3.18e+01$	=	$1.25e+02 \pm 2.00e+01$	=	$\mathbf{7.03e+01 \pm 2.46e+01}$	-
f_{20}	$9.02e+04 \pm 2.84e+04$	$2.27e+05 \pm 9.24e+04$	+	$3.67e+04 \pm 5.73e+03$	-	$\mathbf{1.13e+03 \pm 6.73e+02}$	-
f_{21}	$4.70e+06 \pm 1.79e+06$	$1.04e+08 \pm 4.15e+07$	+	$1.20e+07 \pm 2.51e+06$	+	$\mathbf{3.68e+05 \pm 2.35e+05}$	-
f_{22}	$2.24e+03 \pm 3.44e+02$	$4.79e+03 \pm 5.39e+02$	+	$\mathbf{1.71e+03 \pm 2.88e+02}$	-	$1.84e+03 \pm 3.40e+02$	-
f_{23}	$3.42e+02 \pm 2.73e+00$	$3.45e+02 \pm 2.19e+00$	+	$3.42e+02 \pm 2.69e-01$	=	$\mathbf{3.40e+02 \pm 1.81e-11}$	-
f_{24}	$4.08e+02 \pm 2.66e+01$	$4.08e+02 \pm 1.73e+01$	+	$\mathbf{3.58e+02 \pm 1.11e+00}$	-	$3.77e+02 \pm 4.27e+00$	-
f_{25}	$2.67e+02 \pm 1.89e+01$	$2.77e+02 \pm 1.53e+01$	+	$2.90e+02 \pm 3.72e+00$	+	$\mathbf{2.40e+02 \pm 1.70e+01}$	-
f_{26}	$2.05e+02 \pm 2.09e+00$	$2.01e+02 \pm 4.46e-01$	-	$2.07e+02 \pm 9.80e-01$	+	$\mathbf{2.00e+02 \pm 1.09e-01}$	-
f_{27}	$2.76e+03 \pm 1.30e+02$	$\mathbf{1.55e+03 \pm 1.24e+02}$	-	$1.94e+03 \pm 1.15e+02$	-	$2.26e+03 \pm 1.59e+02$	-
f_{28}	$6.15e+03 \pm 1.24e+03$	$1.44e+04 \pm 9.27e+02$	+	$1.20e+04 \pm 9.41e+02$	+	$\mathbf{5.13e+03 \pm 8.24e+02}$	-
f_{29}	$4.99e+03 \pm 1.08e+03$	$5.30e+04 \pm 4.86e+04$	+	$9.90e+03 \pm 2.10e+03$	+	$\mathbf{2.03e+03 \pm 3.26e+02}$	-
f_{30}	$5.30e+04 \pm 2.34e+04$	$1.84e+06 \pm 1.19e+06$	+	$1.94e+05 \pm 3.80e+04$	+	$\mathbf{1.45e+04 \pm 4.05e+03}$	-

Appendix E. Parameter analysis

We provide a detailed analysis of the parameter settings that characterize the RI component. In particular, we focus our analysis on the crossover logic underlying RI, comparing two variants inherited from Differential Evolution, namely the binomial ("bin") and exponential ("exp") crossover. The first one is the original crossover strategy devised in DE, see the pseudo-code in Algorithm 4; the latter is what we used in our previous experiments, see the pseudo-code in Algorithm 3). This comparison is necessary since it is possible that the exponential crossover contains an implicit bias, due to the fact that the variables which are worth being inherited together might be placed close to each other in the solution encoding: to rule out this possibility, we compare the exponential crossover approach with the (possibly, bias-free) binomial crossover.

For each of the two crossover variants (which, in turn, result in two versions of the RIcDE algorithm that we in this analysis we will refer to as RIcDE-exp and RIcDE-bin, respectively), we tested the algorithmic performance corresponding to various values for the local budget of the compact algorithm (10%, 15%, 20%, 25% and 30%) and for the α parameter (5%, 10%, 15%, 20%, and 25%), for a total of 25 configurations for both the RIcDE-exp and RIcDE-bin variants.

To draw meaningful conclusions w.r.t. the problem properties, we focus the parameter analysis on three representative benchmark functions, namely the Sphere, Rastrigin and Ackley functions[3] in 10, 50 and 100D, defined respectively in $[-5.12, 5.12]^{\mathrm{D}}$, $[-5, 5]^{\mathrm{D}}$ and $[-1, 1]^{\mathrm{D}}$. More specifically, we investigate the effect of modality rather than separability, since all the three functions are separable but the Sphere function is unimodal while the other two are multimodal. This choice was made essentially because, as we have seen, most of CEC2014 problems are non-separable, while the CEC2013-LSGO problems are characterized by different levels of partial separability. Therefore, we focus here only on fully-separable functions, that were somehow underrepresented in the

[3]See `https://en.wikipedia.org/wiki/Test_functions_for_optimization`.

two benchmarks used in the previous experimentation.

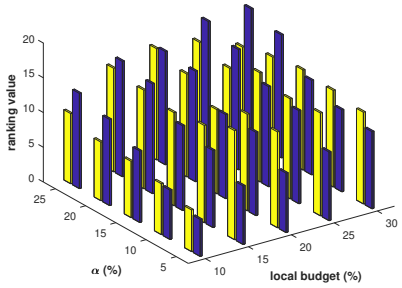

Figure E.5: Holm-Bonferroni Ranks (the higher, the better) for RIcDE-exp (blue) and RIcDE-bin (yellow) evaluated on the Sphere, Rastrigin and Ackley functions in three dimensionalities (10, 50 and 100) at various values for the local budget (10%, 15%, 20%, 25%, and 30%) and for the α parameter (5%, 10%, 15%, 20%, and 25%). A total of 25 configurations for each of the two variants (exp/bin) is shown.

In Figure E.5 we report an aggregate visualization of the Holm-Bonferroni Ranks (the higher, the better) of the 50 configurations of RIcDE-exp (blue) and RIcDE-bin (yellow). The corresponding details of the Holm-Bonferroni procedures are reported in Tables E.25 and E.26, respectively for RIcDE-exp and RIcDE-bin. In the two tables, we use the notation "RIcDEbXXaYYZZZ", where XX is the local budget (%), YY is the α parameter (%), and ZZZ is "exp" or "bin".

In Table E.25, the configuration RIcDEb30a25exp is set as reference (meaning RIcDE with 30% local budget and $\alpha = 25\%$), since it has the highest Rank (this can also be seen in the highest blue bar in Figure E.5). The Holm-Bonferroni procedure shows in this case that most of the parameter configurations are statistically equivalent (null-hypothesis accepted), except for RIcDEb20a5exp, RIcDEb15a5exp, RIcDEb10a10exp and RIcDEb10a5exp. This shows that there are just a few parameter configurations that overall lead to lower performances. An interesting trend, that can be noted looking at the Ranks in both Table E.25 and Figure E.5, is that the α parameter does play a role when the exponential crossover is used since the configurations with higher α values quite consistently Rank higher than the ones with lower values. On the other hand, the indication on the local budget parameter is somehow less clear, although it appears that higher values perform better. In a nutshell, it

Table E.25: Holm-Bonferroni procedure (reference: RIcDEb30a25exp, Rank=1.90e+01) on the Sphere, Rastrigin and Ackley functions in 10, 50 and 100D.

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcDEb25a25exp	1.89e+01	-3.33e-02	4.87e-01	5.00e-02	Accepted
2	RIcDEb30a20exp	1.77e+01	-4.00e-01	3.45e-01	2.50e-02	Accepted
3	RIcDEb25a20exp	1.73e+01	-5.00e-01	3.09e-01	1.67e-02	Accepted
4	RIcDEb15a25exp	1.66e+01	-7.33e-01	2.32e-01	1.25e-02	Accepted
5	RIcDEb20a25exp	1.63e+01	-8.00e-01	2.12e-01	1.00e-02	Accepted
6	RIcDEb20a20exp	1.59e+01	-9.33e-01	1.75e-01	8.33e-03	Accepted
7	RIcDEb15a20exp	1.58e+01	-9.67e-01	1.67e-01	7.14e-03	Accepted
8	RIcDEb25a15exp	1.44e+01	-1.37e+00	8.59e-02	6.25e-03	Accepted
9	RIcDEb10a25exp	1.37e+01	-1.60e+00	5.48e-02	5.56e-03	Accepted
10	RIcDEb30a15exp	1.36e+01	-1.63e+00	5.12e-02	5.00e-03	Accepted
11	RIcDEb25a10exp	1.34e+01	-1.67e+00	4.78e-02	4.55e-03	Accepted
12	RIcDEb10a20exp	1.24e+01	-1.97e+00	2.46e-02	4.17e-03	Accepted
13	RIcDEb15a15exp	1.22e+01	-2.03e+00	2.10e-02	3.85e-03	Accepted
14	RIcDEb20a15exp	1.21e+01	-2.07e+00	1.94e-02	3.57e-03	Accepted
15	RIcDEb20a10exp	1.20e+01	-2.10e+00	1.79e-02	3.33e-03	Accepted
16	RIcDEb30a10exp	1.18e+01	-2.17e+00	1.51e-02	3.13e-03	Accepted
17	RIcDEb15a10exp	1.11e+01	-2.37e+00	8.97e-03	2.94e-03	Accepted
18	RIcDEb30a5exp	1.11e+01	-2.37e+00	8.97e-03	2.78e-03	Accepted
19	RIcDEb10a15exp	1.03e+01	-2.60e+00	4.66e-03	2.63e-03	Accepted
20	RIcDEb25a5exp	9.89e+00	-2.73e+00	3.13e-03	2.50e-03	Accepted
21	RIcDEb20a5exp	8.56e+00	-3.13e+00	8.64e-04	2.38e-03	Rejected
22	RIcDEb15a5exp	8.44e+00	-3.17e+00	7.71e-04	2.27e-03	Rejected
23	RIcDEb10a10exp	7.11e+00	-3.57e+00	1.81e-04	2.17e-03	Rejected
24	RIcDEb10a5exp	5.33e+00	-4.10e+00	2.07e-05	2.08e-03	Rejected

seems that providing larger local budgets with a higher inheritance is the best parameter choice.

Similarly, in Table E.26, where RIcDEb20a25bin is set as the reference algorithm because of its highest Rank, all the configurations are statistically equivalent (i.e. null-hypothesis accepted), with the exception of RIcDEb10a5bin. On

Table E.26: Holm-Bonferroni procedure (reference: RIcDEb20a25bin, Rank=1.60e+01) on the Sphere, Rastrigin and Ackley functions in 10, 50 and 100D.

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	RIcDEb25a15bin	1.57e+01	-1.00e-01	4.60e-01	5.00e-02	Accepted
1	RIcDEb15a5bin	1.57e+01	-1.00e-01	4.60e-01	5.00e-02	Accepted
2	RIcDEb25a25bin	1.52e+01	-2.33e-01	4.08e-01	2.50e-02	Accepted
3	RIcDEb15a25bin	1.49e+01	-3.33e-01	3.69e-01	1.67e-02	Accepted
4	RIcDEb20a20bin	1.48e+01	-3.67e-01	3.57e-01	1.25e-02	Accepted
5	RIcDEb25a5bin	1.48e+01	-3.67e-01	3.57e-01	1.00e-02	Accepted
6	RIcDEb30a15bin	1.44e+01	-4.67e-01	3.20e-01	8.33e-03	Accepted
7	RIcDEb25a10bin	1.44e+01	-4.67e-01	3.20e-01	7.14e-03	Accepted
8	RIcDEb15a20bin	1.42e+01	-5.33e-01	2.97e-01	6.25e-03	Accepted
9	RIcDEb15a10bin	1.40e+01	-6.00e-01	2.74e-01	5.56e-03	Accepted
10	RIcDEb30a10bin	1.39e+01	-6.33e-01	2.63e-01	5.00e-03	Accepted
10	RIcDEb25a20bin	1.39e+01	-6.33e-01	2.63e-01	5.00e-03	Accepted
11	RIcDEb20a10bin	1.39e+01	-6.33e-01	2.63e-01	4.55e-03	Accepted
12	RIcDEb20a5bin	1.38e+01	-6.67e-01	2.52e-01	4.17e-03	Accepted
13	RIcDEb30a20bin	1.38e+01	-6.67e-01	2.52e-01	3.85e-03	Accepted
14	RIcDEb15a15bin	1.34e+01	-7.67e-01	2.22e-01	3.57e-03	Accepted
15	RIcDEb30a5bin	1.32e+01	-8.33e-01	2.02e-01	3.33e-03	Accepted
16	RIcDEb30a25bin	1.28e+01	-9.67e-01	1.67e-01	3.13e-03	Accepted
17	RIcDEb20a15bin	1.21e+01	-1.17e+00	1.22e-01	2.94e-03	Accepted
18	RIcDEb10a25bin	1.01e+01	-1.77e+00	3.86e-02	2.78e-03	Accepted
19	RIcDEb10a20bin	8.44e+00	-2.27e+00	1.17e-02	2.63e-03	Accepted
20	RIcDEb10a15bin	8.11e+00	-2.37e+00	8.97e-03	2.50e-03	Accepted
21	RIcDEb10a10bin	7.33e+00	-2.60e+00	4.66e-03	2.38e-03	Accepted
22	RIcDEb10a5bin	6.00e+00	-3.00e+00	1.35e-03	2.27e-03	Rejected

945 the other hand, differently from the exponential crossover case, in this case the Ranks do not show any clear indication on the effect of the two parameters, as also seen in Figure E.5.

Following this analysis, the question arises as to which of the two best "exp" or "bin" configurations performs best overall. To assess this, we per-

formed a separate comparison of the two reference algorithms for the "exp" or "bin" configurations, namely RIcDEb30a25exp and RIcDEb20a25bin. Table E.27 indicates that the null-hypothesis or statistical equivalence between RIcDEb30a25exp and RIcDEb20a25bin is rejected, highlighting that the best "exp" configuration overall performs better on the three functions in three dimensionalities. The Wilcoxon Rank-Sum test, performed separately for the three dimensionalities (see Tables E.28-E.30) confirms that the best "exp" configuration consistently outperforms the best "bin" configuration. Therefore, we can conclude that, in general, it is recommended to use an exponential crossover for the RI mechanism.

Table E.27: Holm-Bonferroni procedure (reference: RIcDEb30a25exp, Rank=2.00e+00) on the Sphere, Rastrigin and Ackley functions in 10, 50 and 100D.

j	Optimizer	Rank	z_j	p_j	α/j	Hypothesis
1	RIcDEb20a25bin	1.00e+00	-5.20e+00	1.02e-07	5.00e-02	Rejected

To gain further insight into the RI parametrization, we also performed the Holm-Bonferroni procedure independently on each problem and dimensionality (thus, nine procedures in total, i.e. three problems for three dimensionalities). In this case, we omit for brevity the complete Holm-Bonferroni tables but we only compare RIcDE-exp and RIcDE-bin in terms of the average of the best fitness values (the lower, the better) obtained at the end of 30 independent runs per each algorithm and problem, see Figure E.6. This problem-specific analysis reveals some interesting findings. From the figure, it can be seen that on the Sphere and Ackley functions in 10 and 50 dimensions the lowest errors are obtained with the configurations with a local budget greater than or equal to 15%, regardless of the α value, for both RIcDE-exp and RIcDE-bin. This similar behavior on the two functions is somehow surprising since the Sphere function is unimodal while the Ackley function is multimodal. This situation drastically changes though when the dimensionality grows to 100, see the last row in Figure E.6, these same configurations seem to perform worse, with most

of the configurations being somehow equivalent. For all dimensionalities on the Sphere and Ackley functions, we also observe that when the local budget is small (10%) the error increases as α decreases: with many restarts and shorter budgets for each execution of the compact algorithm, it is better to inherit larger portions of the best individual.

Table E.28: Average fitness ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDEb30a25exp) for RIcDEb30a25exp against RIcDEb20a25bin on the Sphere, Rastrigin and Ackley functions in 10D.

	RIcDEb30a25exp	RIcDEb20a25bin	
Sphere	$\mathbf{1.50e-03 \pm 1.15e-03}$	$3.70e-03 \pm 1.59e-03$	+
Ackley	$\mathbf{1.00e-02 \pm 2.58e-03}$	$1.55e-02 \pm 4.01e-03$	+
Rastrigin	$\mathbf{2.27e-01 \pm 2.60e-01}$	$5.31e-01 \pm 2.90e-01$	+

Table E.29: Average fitness ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDEb30a25exp) for RIcDEb30a25exp against RIcDEb20a25bin on the Sphere, Rastrigin and Ackley functions in 50D.

	RIcDEb30a25exp	RIcDEb20a25bin	
Sphere	$\mathbf{8.65e-22 \pm 4.07e-21}$	$4.62e-09 \pm 1.94e-08$	+
Ackley	$\mathbf{4.80e-13 \pm 5.04e-13}$	$1.69e-06 \pm 1.83e-06$	+
Rastrigin	$\mathbf{2.35e+01 \pm 5.25e+00}$	$2.57e+01 \pm 4.08e+00$	=

Table E.30: Average fitness ± standard deviation and Wilcoxon Rank-Sum test (reference: RIcDEb30a25exp) for RIcDEb30a25exp against RIcDEb20a25bin on the Sphere, Rastrigin and Ackley functions in 100D.

	RIcDEb30a25exp	RIcDEb20a25bin	
Sphere	$\mathbf{3.26e+00 \pm 2.46e+00}$	$4.67e+00 \pm 2.11e+00$	+
Ackley	$\mathbf{1.52e-01 \pm 8.46e-02}$	$2.18e-01 \pm 7.57e-02$	+
Rastrigin	$\mathbf{1.62e+02 \pm 2.56e+01}$	$1.81e+02 \pm 1.38e+01$	+

As for what concerns the Rastrigin function, also in this case we observe that at least in 10 dimensions the configurations with a local budget greater than or equal to 15%, regardless of the α value, perform better. On the other hand, for 50 and 100 dimensions all the configurations are practically equivalent in terms of error. The distinct feature of this function compared to the other two is its high multimodality. In fact, the number of local optima is much higher than in the Ackley function. Hence, in this case, there is probably a less clear advantage in inheriting smaller or larger portions of the best solution and allotting smaller or larger local budgets to the compact algorithm.

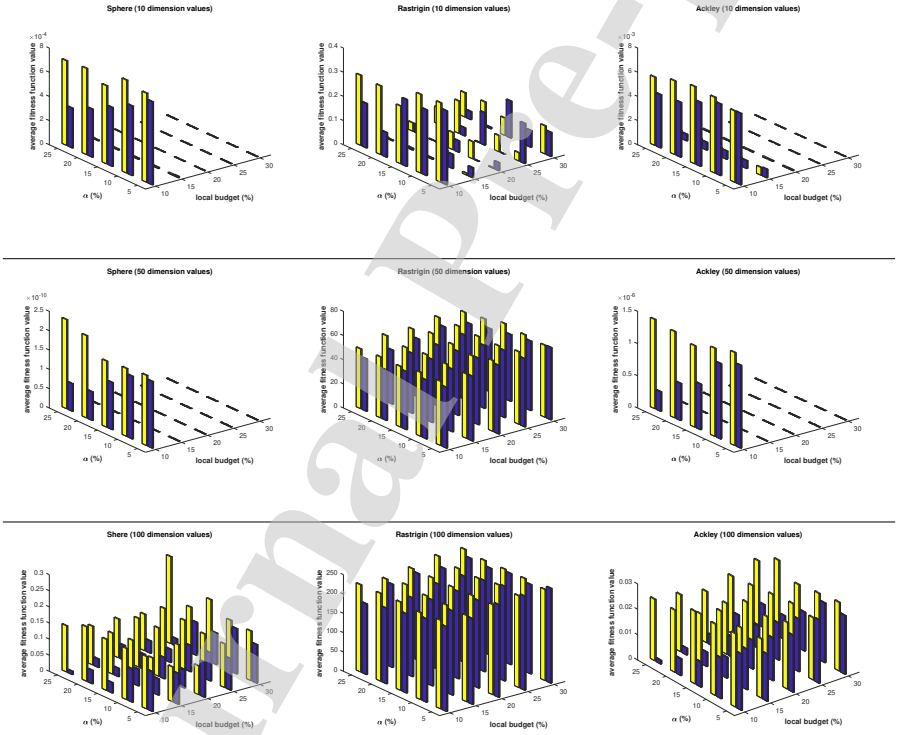

Figure E.6: Average fitness values (the lower, the better) for RIcDE-exp (blue) and RIcDE-bin (yellow) evaluated separately on the Sphere, Rastrigin and Ackley functions in three dimensionalities (10, 50 and 100) at various values for the local budget (10%, 15%, 20%, 25% and 30%) and for the α parameter (5%, 10%, 15%, 20%, and 25%). A total of 25 configurations for each of the two variants (exp/bin) is shown.

Appendix F. Quantitative analysis of convergence

In order to analyze the convergence of the algorithms tested in our experimentation, for each run on each problem and dimensionality we collected the last iteration at which each algorithm obtained an improvement. We consider this value as the *iteration to convergence*. We should note that we adopted this methodology, instead of measuring the number of iterations needed to achieve a given value to reach (i.e., in case of known optimum, to reach the optimal fitness value withing a given threshold ϵ), since in practical applications the optimum is in general unknown. Therefore, this alternative approach is completely problem-agnostic but entirely based on the algorithmic dynamics.

With this consideration in mind, we can analyze Figures F.7, F.8 and F.9, where we report the boxplots (across 30 runs) of the iterations to converge on the CEC2014 in 10D, 50D and 100D respectively. We included in the analysis RIcDE, cDE, the three single-solution algorithms and the three population-based algorithms tested in Section 4, with the same parametrization.

The observation of these boxplots reveals some interesting findings: for instance, in 10D it can be seen that CMA-ES 1+1 tends to converge, on average, quicker than the other algorithms, while, in general, RIcDE tends to converge earlier than cDE (although with a larger variation) and most of the other algorithms. In 50D and 100D, on the other hand, it can be seen that most of the algorithms make use of the entire budget (keeping improving the fitness until the last iteration), but also in this case it should be remarked that RIcDE converges overall earlier than cDE yet with a larger variation.

Figure F.7: Number of iterations to converge on the CEC2014 in 10D for the tested algorithms. The figures, from top-left to bottom right, are related to the benchmark functions from f_1 to f_{30}.

Figure F.8: Number of iterations to converge on the CEC2014 in 50D for the tested algorithms. The figures, from top-left to bottom right, are related to the benchmark functions from f_1 to f_{30}.

Figure F.9: Number of iterations to converge on the CEC2014 in 100D for the tested algorithms. The figures, from top-left to bottom right, are related to the benchmark functions from f_1 to f_{30}.

Highlights

- We present a framework combining a partial restart mechanism with compact algorithms
- We test the framework on a broad range of benchmark and real-world problems
- Different memory-limited algorithms perform differently at different scales
- Compact algorithms with restart can perform better than population-based algorithms

Giovanni Iacca: Conceptualization; Methodology; Software; Writing - original draft. **Fabio Caraffini**: Conceptualization; Methodology; Software; Data curation; Investigation; Formal analysis; Visualization; Validation; Writing - review & editing.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: