Information Sciences 216 (2012) 50–92

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Evolving cognitive and social experience in Particle Swarm Optimization through Differential Evolution: A hybrid approach

M.G. Epitropakis [a,c,*], V.P. Plagianakos [b,c], M.N. Vrahatis [a,c]

[a] Department of Mathematics, University of Patras, GR-26110 Patras, Greece
[b] Department of Computer Science and Biomedical Informatics, University of Central Greece, 2–4 Papassiopoulou Street, GR-35100 Lamia, Greece
[c] Computational Intelligence Laboratory (CI Lab.), Department of Mathematics, University of Patras, GR-26110 Patras, Greece

ARTICLE INFO

Article history:
Received 27 July 2011
Received in revised form 21 May 2012
Accepted 25 May 2012
Available online 12 June 2012

Keywords:
Global Optimization
Particle Swarm Optimization
Differential Evolution
Hybrid approach
Social and cognitive experience
Swarm intelligence

ABSTRACT

In recent years, the Particle Swarm Optimization has rapidly gained increasing popularity and many variants and hybrid approaches have been proposed to improve it. In this paper, motivated by the behavior and the spatial characteristics of the social and cognitive experience of each particle in the swarm, we develop a hybrid framework that combines the Particle Swarm Optimization and the Differential Evolution algorithm. Particle Swarm Optimization has the tendency to distribute the best personal positions of the swarm particles near to the vicinity of problem's optima. In an attempt to efficiently guide the evolution and enhance the convergence, we evolve the personal experience or memory of the particles with the Differential Evolution algorithm, without destroying the search capabilities of the algorithm. The proposed framework can be applied to any Particle Swarm Optimization algorithm with minimal effort. To evaluate the performance and highlight the different aspects of the proposed framework, we initially incorporate six classic Differential Evolution mutation strategies in the canonical Particle Swarm Optimization, while afterwards we employ five state-of-the-art Particle Swarm Optimization variants and four popular Differential Evolution algorithms. Extensive experimental results on 25 high dimensional multimodal benchmark functions along with the corresponding statistical analysis, suggest that the hybrid variants are very promising and significantly improve the original algorithms in the majority of the studied cases.

1. Introduction

The Particle Swarm Optimization (PSO) algorithm is an Evolutionary Computation method, which belongs to the broad class of Swarm Intelligence methods. The PSO algorithm was introduced by Kennedy and Eberhart [20] and is inspired by the social behavior of bird flocking and fish schooling. It is based on a social-psychological model of social influence and social learning. The fundamental hypothesis to the development of PSO is that an evolutionary advantage is gained through the social sharing of information among members of the same species. Moreover, the behavior of the individuals of a flock corresponds to fundamental rules, such as nearest-neighbor velocity matching and acceleration by distance [47,19,12]. The PSO algorithm is capable of handling non-differentiable, discontinuous and multimodal objective functions and has gained increasing popularity in recent years due to its relative simplicity and its ability to efficiently and effectively tackle several real-world applications [10,21]. Without loss of generality, we will consider only minimization problems. In this case, the objective is to locate a global minimizer of a function f (objective function).

* Corresponding author at: Department of Mathematics, University of Patras, GR-26110 Patras, Greece. Tel.: +30 2610 997348.
E-mail addresses: mikeagn@math.upatras.gr (M.G. Epitropakis), vpp@ucg.gr (V.P. Plagianakos), vrahatis@math.upatras.gr (M.N. Vrahatis).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.05.017

Definition 1. A global minimizer $x^\star \in \mathbb{R}^D$ of the real–valued function $f: \mathcal{E} \to \mathbb{R}$ is defined as:

$$f(x^\star) \leqslant f(x), \quad \forall x \in \mathcal{E},$$

where the compact set $\mathcal{E} \subseteq \mathbb{R}^D$ is a D–dimensional scaled translation of the unit hypercube.

To improve the performance and the convergence behavior of Particle Swarm Optimization algorithm, several hybrid approaches have been proposed [73,75,77,49,87,5,2,38,106,99]. One class of variations include hybrids that combine the PSO and the Differential Evolution (DE) algorithms [92]. These approaches aim to aggregate the advantages of both methods to efficiently tackle the optimization problem at hand. The PSO–DE hybrids usually combine the evolution schemes of both algorithms to produce a new evolutionary position scheme [16,68,15,35,71,110]. Specifically, they apply one of the two algorithms as local search to evolve some pre-specified particles [45,113,37], or evolve the control parameters with one of the evolutionary approaches to produce a parameter-free hybrid [68,88,67,24]. Several hybridization perspectives of the PSO algorithm have been proposed in the literature, amongst others the interested reader should refer to the following review papers [106,99]. Thangaraj et al. [99], review the hybridization perspectives of the PSO algorithm with various different algorithm such as Genetic Algorithms, Differential Evolution and other techniques. Consequently, Xin et al. [106] present a thorough review of the state-of-the-art hybridization perspectives of PSO and DE algorithms, as well as several other representative hybrids. They provide classification mechanisms and a comprehensive taxonomy to differentiate and analyze the existing hybrid PSO/DE algorithms.

The aforementioned hybrid approaches can be viewed under the general concept of Memetic Computing (MC) [69,59,62,63]. As stated in [62], "*Memetic Computing is a broad subject which studies complex and dynamic computing structures composed of interacting modules (memes) whose evolution dynamics is inspired by the diffusion of ideas. Memes are simple strategies whose harmonic coordination allows the solution of various problems*". In other words, complex ideas can be analyzed as memes which propagate, interact and change within a population. Thereby, their character will constantly undergo evolution and tend towards progressive improvements. In Computational Intelligence a meme can be identified as an agent, a search strategy, an operator, or a complex system component, e.g. an optimization methodology which constantly evolves through time. The research field of MCs has blossomed in the last decade resulting in many successful methodologies. In general, the importance of MC methodologies is related to the No Free Lunch theorem (NFL) [105]. The NFL theorem mathematically proves that the average performance of any pair of optimization algorithms, across all possible optimization problems is identical. Thereby, an optimization algorithm which performs well on a class of problems, will certainly perform worse on a set of the remaining optimization problems, since this is the only possibility in which the performance of the pair of algorithms will be on average equal over all optimization problems. Based on this theorem, the broad class of MC methodologies became a common practice in the general field of Computational Intelligence and particular in the Evolutionary Computing community, i.e. the combination of various interacting algorithmic components (memes) to efficient tackle a specific class of problems.

Similarly, under the general concept of MC as a composition of interacting/evolving modules (memes), we propose a general and simple framework to hybridize the well-known PSO and DE optimization algorithms. Specifically, the current study has been motivated by the behavior and the spatial characteristics of the personal experience of each particle (*memory swarm*) of the PSO algorithm, during the evolution process. PSO incorporates in its swarms two main concepts, the *explorer-swarm* and the *memory-swarm* [10]. During evolution each particle in the swarm remembers its current position in the search space, as well as the best position it has ever encountered so far, i.e. the so called personal best position. Thereby PSO explores the most promising regions of search space (*explorer-swarm*), while in parallel retains the best positions found so far by the swarm (*memory-swarm*). The movement of each particle is controlled by two forces, related to the best previous position of the particle (cognitive experience) and the position attained by the best particle in its society (social experience), i.e. either the whole swarm or its neighborhood. Extensive simulations indicate that through the evolution process of the PSO algorithm, the cognitive experience of each particle, i.e. the *memory-swarm*, tend to be distributed in the vicinity of the problem's optima. To this end, we propose a hybrid evolutionary framework to efficiently evolve the social and cognitive experience of the swarm and enhance the convergence properties of the PSO algorithm. Here, we incorporate the Differential Evolution algorithm as the second interactive module, which is a simple and compact evolutionary algorithm exhibiting good convergence characteristics. To evaluate the performance of the proposed framework, we initially apply six classic DE mutation strategies on the canonical PSO. Afterwards, we combine five state-of-the-art PSO variants with the three best performing DE mutation strategies along with four popular DE algorithms. Extensive experimental results on 25 difficult benchmark functions along with the corresponding statistical analysis suggest that the proposed framework is very promising.

Preliminary results have been presented in [25], in which a part of the *memory-swarm* has been evolved during evolution, in an attempt to improve the convergence characteristics of the PSO algorithm and to reduce any computational overhead of the DE algorithm. The utilized heuristic rule selects to evolve only the best personal position that has been changed (improved) during the previous evolution step. To eliminate the need for any heuristic rule and make the framework even simpler, in the paper at hand we evolve all personal best positions of the swarm. Furthermore, based on the hybrid evolutionary algorithm taxonomy presented in [106], our preliminary work [25], and its extensions in the current study belong to the collaboration-based hybrid PSO/DE approaches. Collaboration indicates that the parent optimizers cooperate with each other in the optimization search space in an attempt to seek the optimum solution. As such, they share or exchange accumulated information during their search operations, while their dynamics are maintained [106].

The novelty of this study lies on the simple structure of the proposed framework and its scalability. The majority of the proposed PSO/DE hybrids incorporate complex rules on several aspects of the hybrid algorithmic structure, resulting in not easily implemented hybrid schemes. Here, we propose to apply an evolutionary algorithm and in particular a DE variant, after each PSO evolution step on the *memory-swarm*. The DE algorithm will evolve the *memory-swarm* as its population, without selecting any specific number of personal best positions to be applied, or incorporating any complex DE/PSO update rule to evolve the best positions. Furthermore, DE is applied on each generation after the PSO evolution operations, without incorporating any heuristic rule to determine the frequency of its application. On the contrary, several approaches are evolving the best personal positions of PSO, but incorporate curious structures, which cannot be easily generalized. For example, in [113] the authors evolve the particles one generation with PSO and one with DE, while they utilize only the DE/rand/2/bin strategy with randomly chosen best positions and as a base vector the best position of the *explorer-swarm*; in [95] the authors implement the previously approach to handle multimodal image registration problems; in [37] DE is applied to evolve the *explorer-swarm* at pre-specified intervals; Pant et al. [70], propose a two phase hybrid that consists of alternating phases of the DE and PSO algorithms; Liu et al. [54] evolve only the half particles of the swarm with PSO, and apply DE on each best position by using the positions of the *explorer-swarm* in the mutation strategies; in [110] the authors update the particles of the swarm with two strategies, namely the DE updating strategy and a combined DE/PSO updating strategy.

In addition, the structure of the proposed framework is extensible, i.e. it can be easily generalized to produce a family of PSO/DE or other PSO-based hybrids. This can be implemented by applying any PSO variant as base algorithm and any DE variant or other evolutionary algorithm to evolve the *memory-swarm* on each generation after the PSO evolutionary operation. As previously mentioned, in the current study we straightforwardly apply the proposed framework on the canonical PSO and five PSO variants with different dynamics. In addition, the proposed framework is implemented with six classic DE mutation strategies and four popular DE algorithms. The interested reader should only consider about the characteristics of the applied algorithms and their effects on the performance of the resulting scheme.

The rest of this article is organized as follows: Section 2 briefly describes the basic operations of the canonical PSO and the DE algorithms, and presents several related methodologies. In Section 3, we analyze the behavior of the cognitive and social experience in the PSO algorithm that motivated the proposed approach. In Section 4, we propose the new hybrid evolutionary framework and discuss its characteristics, while in Section 5 we present an extensive experimental analysis. The paper concludes in Section 6, with a discussion and some pointers for future work.

2. Background material

For completeness purposes, in this section we briefly describe the basic operations as well as some insights about the main characteristics of the canonical Particle Swarm Optimization and the Differential Evolution algorithms. The section ends with a literature review of several recently proposed PSO variants.

2.1. The Particle Swarm Optimization algorithm

The PSO algorithm is a population-based stochastic algorithm that exploits a population of individuals to effectively probe promising regions of the search space. PSO incorporates two main concepts in its swarms, the *explorer-swarm* and the *memory-swarm* [10]. During its execution each individual (*particle*) of the population (*swarm*) moves with an adaptable velocity within the search space in an attempt to explore the most promising regions (*explorer-swarm*) and simultaneously retains in its memory/experience the best position it ever encountered (*memory-swarm*). There exist two main PSO versions; namely the *global* PSO and the *local* PSO. In the *global* PSO version, the best position ever attained by the individuals of the swarm is communicated to all the particles. On the other hand, the *local* PSO versions incorporate into the PSO structure the neighborhood concept, in which the best position of each particle can be propagated between neighboring particles. The neighborhoods can be either static with pre-specified topologies or dynamic with variable or adaptive topologies, while they may utilize either spatial or network-based characteristics [19,57,56,6,46,77,32,11].

More specifically, each particle is a D-dimensional vector and the swarm S consists of NP particles, i.e. if g is the current time step (*generation*) then the swarm can be denoted as $S_g = \{X_g^1, X_g^1, \ldots, X_g^{NP}\}$. Therefore, the position of the i-th particle of the swarm can be represented as: $X_g^i = (x_g^{i,1}, x_g^{i,2}, \ldots, x_g^{i,D})$. The velocity of each particle is also a D-dimensional vector and for the i-th particle is denoted as: $V_g^i = (v_g^{i,1}, v_g^{i,2}, \ldots, v_g^{i,D})$. The best previous position of the i-th particle can be recorded as: $P_g^i = (p_g^{i,1}, p_g^{i,2}, \ldots, p_g^{i,D})$, the best particle in the swarm (i.e. in minimization problems, the particle with the smallest fitness function value) is indicated by P_g^{best}, while the best particle in the neighborhood of the i-th particle as $P_g^{\text{best}_i}$. Furthermore, the neighborhood of each particle is usually defined through its index. The majority of the PSO variants utilize the *ring* topology, which is the most common topology in the literature. In the *ring* topology, the neighborhood of each particle consists of particles with neighboring indices [6,57,32]. Nevertheless, other topologies have been also studied [19,57,56,6,46,77,32].

In the present study, we consider the canonical PSO version proposed by Clerc and Kennedy [12], which incorporates the parameter χ, known as the *constriction factor*. The main role of the constriction factor is to control the magnitude of the velocities and alleviate the "swarm explosion" effect that sometimes prevented the convergence of the original PSO

algorithm [1,12]. As stated in [12], for each time step g (*generation*), the dynamic behavior of the particles in the swarm is manipulated using the following equations:

$$V_{g+1}^i = \chi\left(V_g^i + c_1 r_1\left(P_g^i - X_g^i\right) + c_2 r_2\left(P_g^{\text{best}_i} - X_g^i\right)\right), \tag{1}$$

$$X_{g+1}^i = X_g^i + V_{g+1}^i, \tag{2}$$

for $i = 1, 2, \ldots, NP$, where χ is the constriction factor parameter, c_1 and c_2 are positive constants referred to as *cognitive* and *social* parameters respectively and r_1 and r_2 are randomly chosen numbers uniformly distributed in [0, 1]. The cognitive parameter controls the experience influence of each particle with respect to its best performance found so far, while the social parameter with respect to the best position found by its society, i.e. either the whole swarm or its neighborhood. Furthermore, in a stability analysis provided in [12], it was implied that the constriction factor is typically calculated according to the following formula:

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \tag{3}$$

where $\phi = c_1 + c_2$ and $\kappa = 1$, and to guarantee the quick convergence of the scheme the value of ϕ has to satisfy $\phi > 4$. The aforementioned scheme is typically utilized for the constant $\phi = 4.1$, with $\chi = 0.72984$ and $c_1 = c_2 = 2.05$ [6,12].

The next section briefly describes the basic operators of the Differential Evolution algorithm.

2.2. The Differential Evolution algorithm

The DE algorithm [92,83,17,80,27,65] is a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of Evolutionary Algorithms (EAs). The DE method requires few control parameters and several experimental studies have shown that DE has good convergence properties and outperforms other well known and widely used EAs [92,8,83,101,17,65].

More specifically, DE is a population–based stochastic algorithm that exploits a population of potential solutions (*individuals*) to effectively probe the search space. Like PSO, the population of individuals is randomly initialized in the optimization domain with *NP*, *D*-dimensional, vectors following usually a uniform probability distribution. Individuals evolve over successive iterations to explore the search space and try to locate the minima of the objective function. Throughout the execution process, the user-defined population size, *NP*, is fixed. At each iteration, called *generation*, new vectors are derived by the combination of randomly chosen vectors from the current population. This operation in our context can be referred to as *mutation*, while the outcoming vectors as *mutant individuals*. Each mutant individual is then mixed with another, predetermined, vector – the *target vector* – through an operation called *recombination*. This operation yields the so-called *trial vector*. Finally, the trial vector undergoes the *selection* operator, according to which it is accepted as a member of the population of the next generation, only if it yields a reduction in the value of the objective function f relative to that of the target vector. Otherwise, target vector is retained in the next generation. The search operators efficiently shuffle information among the individuals, enabling the search for an optimum to focus on the most promising regions of the solution space.

Below, we describe the original mutation operators proposed in [92]. Specifically, for each individual x_g^i, $i = 1, \ldots, NP$, where g denotes the current generation, the mutant individual v_{g+1}^i can be generated according to one of the following equations:

1. DE/best/1

$$v_{g+1}^i = x_g^{\text{best}} + F(x_g^{r_1} - x_g^{r_2}), \tag{4}$$

2. DE/rand/1

$$v_{g+1}^i = x_g^{r_1} + F(x_g^{r_2} - x_g^{r_3}), \tag{5}$$

3. DE/current-to-best/1

$$v_{g+1}^i = x_g^i + F(x_g^{\text{best}} - x_g^i) + F(x_g^{r_1} - x_g^{r_2}), \tag{6}$$

4. DE/best/2

$$v_{g+1}^i = x_g^{\text{best}} + F(x_g^{r_1} - x_g^{r_2}) + F(x_g^{r_3} - x_g^{r_4}), \tag{7}$$

5. DE/rand/2

$$v_{g+1}^i = x_g^{r_1} + F(x_g^{r_2} - x_g^{r_3}) + F(x_g^{r_4} - x_g^{r_5}), \tag{8}$$

where x_g^{best} is the best member of the previous generation, $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, i-1, i+1, \ldots, NP\}$ are random integers mutually different and not equal to the running index i, and $F > 0$ is a real parameter, called *mutation or scaling factor*. The *mutation factor F*, controls the amplification of the difference between two individuals and is used to prevent the risk of stagnation of the search process. It is also mainly responsible for the convergence rate of the algorithm. Therefore, an

inappropriate mutation factor value can cause deceleration of the algorithm and decrease of the population's diversity. In the original DE algorithm, the mutation factor F is a fixed and user–defined parameter, while in many adaptive DE variants each individual is associated with a different adaptive mutation factor [65,108,112,7,103,63,104]. It is clear that more mutation operators can be generated by either using the above ones as building blocks [78] or by incorporating population's spatial information in the above strategies [27]. Several DE variants which either introduce new mutation strategies or new self-adaptive techniques to tune the control parameters have been recently proposed [83,8,29,23,7,112,111,103,63,104,114,31,43].

Furthermore, here we utilize the trigonometric mutation operator [29], which performs a mutation according to the following equation, with probability τ_μ:

6. TDE/rand/1

$$v^i_{g+1} = (x^{r1}_g + x^{r2}_g + x^{r3}_g)/3 + (p_2 - p_1)(x^{r1}_g - x^{r2}_g) + +(p_3 - p_2)(x^{r2}_g - x^{r3}_g) + (p_1 - p_3)(x^{r3}_g - x^{r1}_g), \tag{9}$$

and with probability $(1 - \tau_\mu)$, the mutation is performed according to Eq. (5), where τ_μ is a user defined parameter, typically set around 0.1. The values of p_m, $m = \{1, 2, 3\}$ and p' are obtained through the following equations: $p_1 = \left|f(x^{r1}_g)\right|/p', p_2 = \left|f(x^{r2}_g)\right|/p', p_3 = \left|f(x^{r3}_g)\right|/p'$, and $p' = \left|f(x^{r1}_g)\right| + \left|f(x^{r2}_g)\right| + \left|f(x^{r3}_g)\right|$.

Having performed mutation, the recombination operator is subsequently applied to further increase the diversity of the population. To this end, the mutant individuals are combined with other predetermined individuals, called the target individuals. Specifically, for each component $l\,(l = 1, 2, \ldots, D)$ of the mutant individual v^i_{g+1}, we randomly choose a real number r in the interval $[0, 1]$. Then, we compare this number with the user-defined *recombination constant*, *CR*. If $r \leqslant CR$, then we select, as the l-th component of the trial individual u^i_{g+1}, the l-th component of the mutant individual v^i_{g+1}. Otherwise, the l-th component of the target vector x^i_g becomes the l-th component of the trial vector. This operation yields the trial individual. It is evident that if the value of the recombination constant is too small (close to zero) the effect of the mutation operator is canceled, since the target (and not the mutant) vector will become the new trial vector.

Finally, the trial individual is accepted for the next generation only if it reduces the value of the objective function (selection operator):

$$x^i_{g+1} = \begin{cases} u^i_{g+1}, & \text{if } f(u^i_{g+1}) < f(x^i_g) \\ x^i_g, & \text{otherwise} \end{cases}. \tag{10}$$

2.3. Related work

To improve the performance and the convergence characteristics of the Particle Swarm Optimization algorithm, several variations and hybrid approaches, with altered search dynamics, have been proposed [72,73,77,49,87,5,2,38]. However, some PSO variants have gained a lot of attention and acceptance through the Evolutionary Computing research community, due to their successful novel PSO schemes, which have exhibited great performance gains over a multitude different applications. They have successfully exploited different aspects of the PSO algorithm, either by utilizing novel velocity update rules that enhance PSO's exploratory/exploitative search power or by incorporating in PSO special schemes to exploit the structure of the benchmark function at hand. Representative examples of the former include fully informed topologies [57], barebones velocity updates [46], multiple swarms [4,53], unified operators [77], and comprehensive learning schemes [52], while examples of the latter include schemes such as cooperation and coevolution [3,51].

More specifically, in an attempt to improve the performance of the canonical PSO [47,12] and to simplify its scheme by eliminating its control parameters, Kennedy in [46] proposed the barebones PSO (BBPSO). BBPSO updates its velocity through a Gaussian distribution, where its location parameters depend on the best personal and the best neighborhood positions, i.e. the mean value is the average of these positions, while the variance equals with their distance. Thereby, BBPSO initially facilitates exploration of the search space, since the best personal positions are far away from each other. As the algorithm evolves through time, the deviation of the best personal positions decreases and it focuses on exploiting the average of the best personal and best neighborhood positions. In [57], Mendes et al. proposed a simple strategy to update the positions of each particle, namely the fully informed particle swarm (FIPS). FIPS updates each particle's position with a weighted sum of all of its neighboring particles. Thus, each particle is influenced and attracted by the information of all of its topological neighbors and the best neighboring particle. In addition, Peram et al. proposed in [79] a new algorithm that updates the particle's velocity through observation of three different particles, the personal best, the global best, and the particle that has a higher fitness and is nearer to the current particle with a maximal fitness-to-distance ratio. In turn, the authors in [75–77] tried to tackle the trade-off between the explorative and exploitative characteristics of the canonical PSO by proposing a unified procedure, namely the Unified PSO (UPSO). UPSO utilizes a velocity update scheme that efficiently combines the local and global versions of the canonical PSO algorithm, resulting in a very competitive PSO variant [77]. Moreover, DMSPSO [53] adopts multiple small swarms with a random regrouping strategy to introduce a dynamically

changing neighborhood structure to each particle. To avoid stagnation and enhance the diversity of the swarms, once in a while it randomly regroups the swarm's neighborhoods. Consequently, CLPSO updates each particle's velocity by utilizing a novel learning scheme based on the swarm's best personal positions [52]. It uses the best positions of the other particles as exemplars to be learned from. This strategy produces a scheme on each dimension in which each particle may potentially learn from different exemplars per dimension. Hence, CLPSO makes the particles have more exemplars to learn from and a potential larger search space to roam. CLPSO has exhibited great performance gains mostly on multimodal functions, while it is not the best choice for solving unimodal problems [52].

Furthermore, to tackle large-scale optimization problems, many researchers have employed the cooperation and coevolution concepts in their optimization algorithms [81,3,51,107]. Several works incorporate the inspiring cooperative/coevolution scheme of Potter and De Jong [81]. In PSO, an early attempt to apply Potter's scheme has been made in [3], resulting in two new cooperative schemes, namely CPSO-S_K and CPSO-H_K. The latter scheme is an exact implementation of Potter's scheme to PSO, while the former combines the PSO and the CPSO-S_K algorithms. The resulting algorithms significantly improve the original PSO, due to the cooperation/coevolution of multiple swarms which simultaneously optimize different components of the solution vectors. Another representative work is presented in [51], where a new cooperative coevolving PSO variant is introduced. The proposed scheme has been built on the aforementioned cooperative schemes, while it employs an effective variable grouping technique (random grouping). Additionally, it adopts a new position update rule, which is based on the Cauchy and Gaussian distributions and a coevolving scheme that dynamically determines the variable's subcomponent sizes.

Another active research trend in the last years is the integration of well established and effective PSO variants in new adaptive or self-adaptive schemes, in an attempt to aggregate their characteristics and their search dynamics. To this end, Frankenstein's PSO [61] integrates three distinct algorithmic PSO components to combine their effects and produce an effective optimizer. Specifically, it integrates as first component a time-varying population topology that reduces its connectivity over time to improve the tradeoff between speed and quality associated with topologies of different connectivity degrees. Secondly, based on the good performance gains of the FIPS algorithm compared with other PSO variants with different topologies, it utilizes FIPS as the main velocity update rule. Finally, to tune the exploration and exploitation behavior of the algorithm, the decreasing inertia weight procedure is included as the third component. The resulting scheme is very promising and in many cases it performs better than its three main components. In turn, Wang et al. in [102] proposed a self-adaptive learning-based PSO scheme, which simultaneously adopts four PSO search strategies with a different behavior based on the problem's characteristics. To adapt the four strategies, it incorporates a probabilistic model that tunes the probability of each strategy based on their ability to generate fitter solutions though the optimization procedure.

Furthermore, the concept of heterogeneous PSO variants has been adopted in many recent works [66,60,90,22]. This concept includes either sub-swarms that utilize different meta-heuristic algorithms and cooperate with each other, or swarms with different strategies per particle that are selected from a pool of strategies with a predefined or a dynamic way. Furthermore, inspired by the adaptive filter and the statistical learning theory an adaptive PSO scheme (MultiPSO) has been recently proposed [28]. MultiPSO integrates several different PSO schemes in an attempt to aggregate their characteristics and their search dynamics. Its framework is based on tracking the parameters of a multinomial distribution to capture successful evolution changes of each PSO scheme in the evolutionary process.

3. Studying the cognitive and social experience of PSO

In this section, we investigate the behavior and the dynamics of a swarm in PSO during its evolution. Our findings suggest that the particle's best positions tend to gather around minimizers of the objective function. This behavior motivates our approach, which aims to evolve this knowledge through an evolutionary algorithm. The main goal is to efficiently guide the best knowledge of each particle towards a global optimum, without destroying the search capabilities of the PSO algorithm.

Numerous PSO variations have been proposed to improve the accuracy of solutions and PSO convergence behavior [74,75,57]. In [2,12] it has been formally proven that each particle converges to a weighted average of its best personal and best neighborhood positions. Motivated by this finding, several PSO variants have been introduced that incorporate knowledge, exploiting the best personal positions [46,57]. Moreover, the exploitation of the best personal experience has been incorporated in several PSO variants with many different methodologies. Specifically, some variants adapt the best personal positions using distributions that are based on them (e.g. Barebones PSO [46]) or include their weighted sum (e.g. FIPS [57]). Other variants incorporate update schemes that utilize information of the best personal positions by means of an average of two or more [113,95].

The aforementioned approaches and their convergence characteristics enhance our findings. Extensive experimental simulations have verified that the PSO algorithm tends to distribute the best positions encountered by the particles in the swarm to the vicinity of problem's minima. Additionally, the local version of PSO has more explorative characteristics and tends to distribute the best personal positions to regions around many minima, while the global version of PSO exhibits more exploitive characteristics and rapidly gathers the best personal experience to the basin of attraction of a (global or local) minimum.

To demonstrate this behavior, we will utilize as a case study the two-dimensional Shekel's Foxholes benchmark function. Shekel's Foxholes function can be defined according to the following equation:

$$f(x) = \frac{1}{0.002 + \psi_1(x)},\ x_j \in [-65.536, 65.536],$$

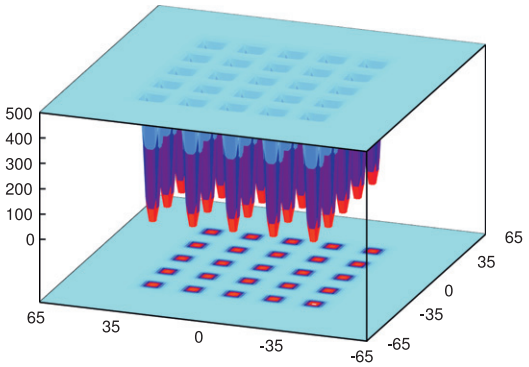

Fig. 1. 3-D plot of the Shekel's Foxholes function.

where $\psi_1(x) = \sum_{i=1}^{25} 1/(i + \sum_{j=1}^{2}(x_j - a_{ij})^6)$. The parameters for this function are:

$$a_{i1} = \{-32, -16, 0, 16, 32\}, \text{ where}$$
$$i = \{0, 1, 2, 3, 4\} \text{ and } a_{i1} = a_{i \bmod 5,1}$$
$$a_{i2} = \{-32, -16, 0, 16, 32\}, \text{where}$$
$$i = \{0, 5, 10, 15, 20\} \text{ and}$$
$$a_{i2} = a_{i+k,2},\ k = \{1, 2, 3, 4\},$$

which has 24 distinct local minima and one global minimum $f(-32, 32) = 0.998004$. To further elucidate Shekel's Foxholes shape, a surface plot is illustrated in Fig. 1.

Additionally, we illustrate the positions of a swarm, as well as the best position of each particle, evolved by the canonical PSO algorithm on the Shekel's Foxholes benchmark function. Specifically, Fig. 2 illustrates contour plots of the Shekel's Foxholes function and the positions of a swarm consisting of 40 particles that have been evolved with the local version of the canonical PSO after 1, 5, 10, and 20 generations, while Fig. 3 demonstrates the distribution of their best personal experience. The two figures show that the role of swarm's positions is to initially explore the search space before gathering them around their attractor's, i.e. the personal and best neighborhood positions. In turn, the swarm's best positions tend to rapidly gather around the basins of attraction of the local/global minima and exploit their regions. It is evident that an efficient strategy to adapt or evolve the swarm's best positions, i.e. the social and cognitive experience of the swarm, may enhance the performance of the original PSO scheme.

To study and evaluate the *clustering tendency* of swarm's components, i.e. the positions and the social and cognitive experience, we utilize a statistical test called the Hopkins test [39]. Clustering tendency is well know concept in the cluster analysis literature and deals with the problem of determining the presence or absence of a clustering structure in a data set [100,44].

The Hopkins test relies on the distances between the vectors of the data set $X = \{x_i, i = 1, 2, \ldots, NP\}$, i.e. the current population and a number of vectors that are randomly placed in the search space. More specifically, let $X\prime = \{y_i, i = 1, 2, \ldots, M\}, M \ll NP$, with typically $M = 0.1 \cdot NP$, be a set of vectors that are uniformly distributed in the search space. In addition, let $X_1 \subset X$ be a set of M randomly chosen vectors of X. Let d_j be the distance of $y_j \in X'$ to its closest vector $x_c \in X_1$, and δ_j be the distance from x_c to its closest vector in $X_1 \setminus \{x_c\}$. Then, the Hopkins statistic involves the k-th powers of d_j and δ_j and it is defined as follows [44,100]:

$$h = \frac{\sum_{j=1}^{M} d_j^k}{\sum_{j=1}^{M} d_j^k + \sum_{j=1}^{M} \delta_j^k}. \tag{11}$$

This statistic compares the nearest neighbor distribution of the points in X_1 with that of the points in X'. When the dataset X contains clusters, the distances between nearest neighbor points in X_1 are expected, on the average, to be small. Consequently, the values of h are relatively large. Notice that large values of h indicate the presence of a clustering structure in the dataset X, while small values of h indicate the presence of regularly spaced points. A value around 0.5 indicates that the vectors of the dataset X are randomly distributed over the search space.

In Fig. 4, we illustrate the mean value of the H-measure at each generation, obtained from 100 independent simulations for the 30-dimensional versions of the first six functions of the CEC'2008 Special Session on Large Scale Global Optimization [96]. These benchmarks were chosen to investigate the behavior of the PSO algorithm in many qualitatively different problems. The function set includes shifted versions of two unimodal and four highly multimodal functions. Error bars around the mean depict the standard deviation of the H-measure. Due to the stochastic nature of H-measure, for every generation in every simulation we calculate the H-measure value 100 times, by considering different random solutions.

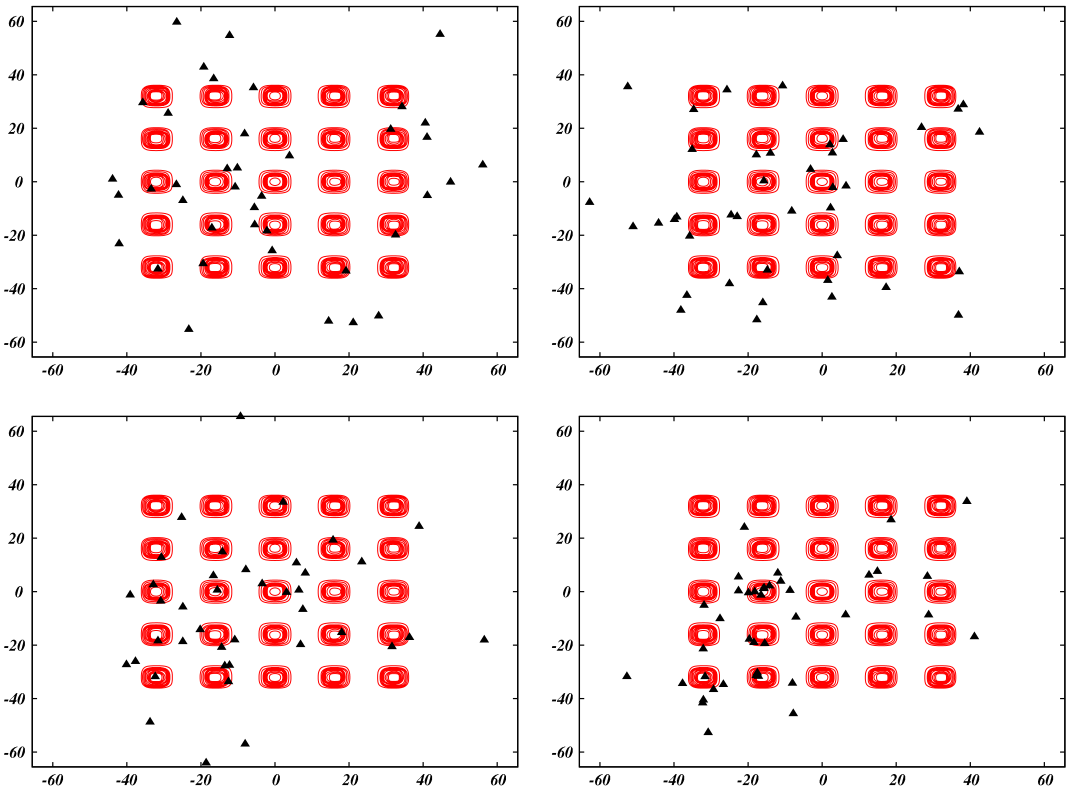

Fig. 2. Local PSO population's positions after 1, 5, 10, and 20 generations.

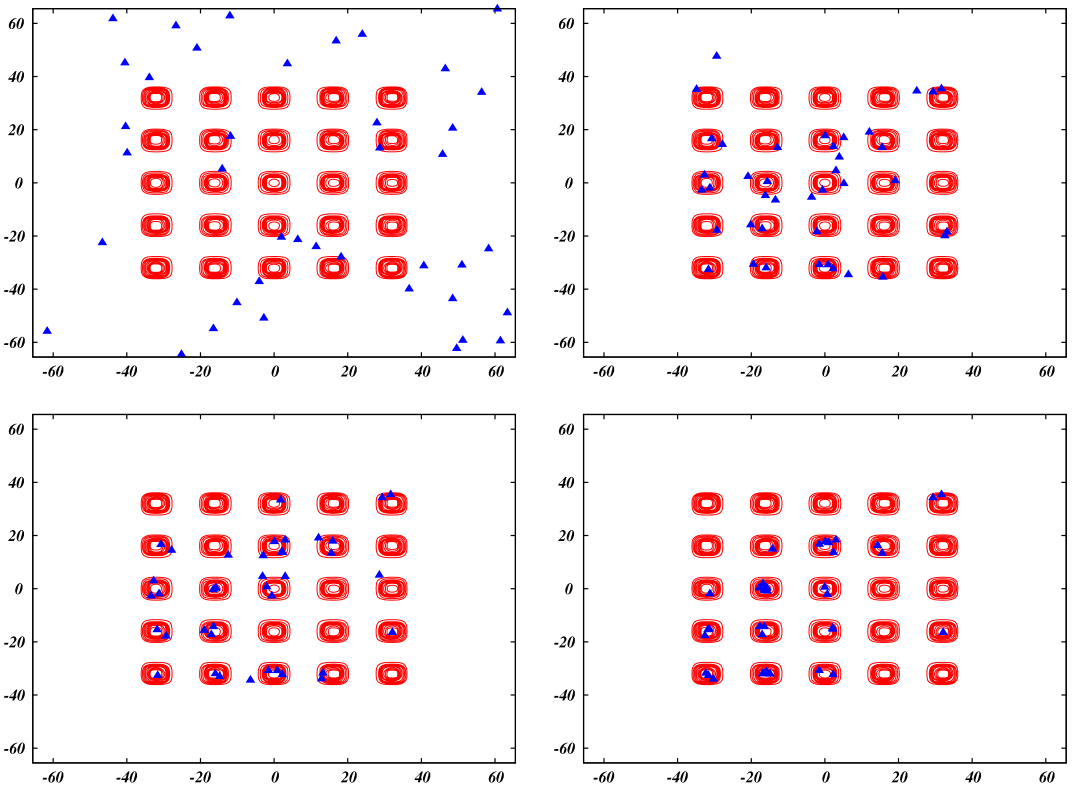

Fig. 3. local PSO population's best personal positions after 1, 5, 10, and 20 generations.

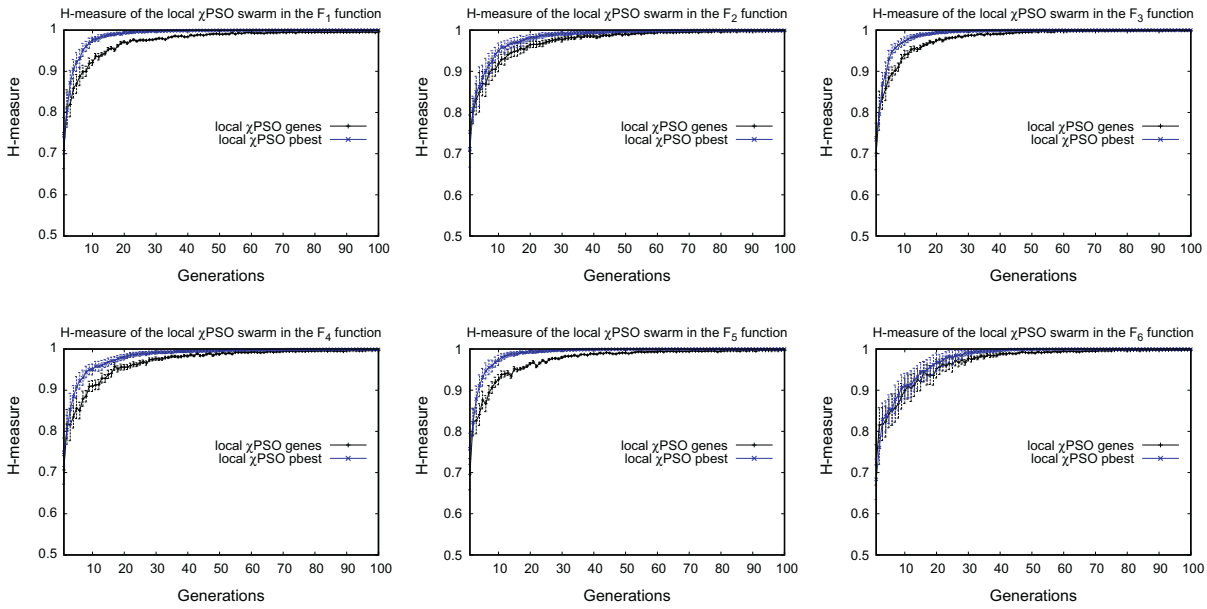

Fig. 4. H-measure of the local χPSO for the six shifted benchmark functions.

As shown, both the positions and the best personal positions exhibit large H-measure values within the first 100 generations, indicative of a strong clustering structure, even in these initial stages of the evolution. Also, the relative values of the H-measure for the different positions indicate an ordering with respect to their exploitation tendency. The best personal positions appears to have an exploitative behavior, while the positions of the swarm seems to exhibit more explorative nature.

In this work, we attempt to take advantage of this clustering behavior. To this end, we propose to evolve the best personal positions of the swarm with the Differential Evolution algorithm. More specifically, after the evolution of a particle we propose to additionally evolve the best personal positions, as well. This will efficiently evolve the social and cognitive experience of the swarm, i.e. the *memory-swarm*, which has the potential either to locate better regions around problem's minima or to rapidly exploit the regions of the already found minima.

4. The proposed framework

Motivated by the aforementioned PSO behavior and our findings, it is possible to guide the evolution towards a global optimum without compromising the algorithm's search dynamics by evolving the best experience of the swarm (social and cognitive) with an intelligent optimization procedure, such as the Differential Evolution algorithm. In this section, we discuss the main concepts behind the hybrid framework of the Particle Swarm Optimization and the Differential Evolution algorithm.

To evolve the social and cognitive experience of a swarm we can utilize several different subpopulations for the evolutionary process, e.g. the positions of the swarm, the best personal positions of the swarm or both of them. Based on their spatial positions the evolutionary process will either explore unexplored regions or exploit the already found ones. Here, we propose to use as population for the evolutionary process, the set of the best personal experience of the swarm (*memory-smarm*), i.e. both the social and the cognitive experience. Thereby, the newly evolved positions will be the outcome of the best experience of the swarm, which will have the potential to either locate better regions around problem's minima or to rapidly exploit the already found regions, and thus accelerate convergence.

4.1. The algorithmic scheme

Let us define the best personal experience set (*memory-smarm*) as $S_g^p = \{P_g^1, P_g^2, \ldots, P_g^{NP}\}$ containing the best personal position of each particle at the gth time step (generation). Hence, after each time step of the PSO algorithm, we apply one DE step to the particles in the S_g^p set. Specifically, the three main DE evolution steps (mutation, recombination, and selection) are applied to the best personal positions. In the mutation procedure one of the aforementioned mutation strategies (Eqs. (4)–(9)) is utilized. Afterwards, we utilize a one-to-one tournament selection between the previous best position and the evolved one, i.e. if the resulting new best position is fitter than the old one then it is accepted for the next generation; otherwise the old best position is retained in the S_g^p set. It is worth noting that the one-to-one tournament selection is

naturally applied in the proposed framework, since it is the selection procedure of the original DE algorithm. A detailed algorithmic scheme of the proposed approach which particular utilizes the canonical PSO variant with a ring neighborhood topology along with the DE/rand/1/bin strategy for evolving the best personal memory (PSO:DE:rand/1/bin) is illustrated in Algorithm 1.

Algorithm 1. The proposed approach PSO:DE:rand/1/bin: the local PSO version algorithmic scheme utilizing a ring topology along with the DE/rand/1/bin algorithm for evolving the best personal experience

---

/* **Initialize particles in the swarm:** $S_{g=0} = \{X^1_{g=0}, X^2_{g=0}, \ldots, X^{NP}_{g=0}\}$, uniformly in the optimization search hyper-rectangle $[L, U]$, */
for $i = 1$ to NP **do**
  for $j = 1$ to D **do**
    $X^{i,j}_{g=0} = L_j + \text{rand}_j(0,1) \cdot (U_j - L_j)$
  end for
  Evaluate Particle $X^i_{g=0}$
end for
for each time step g (*generation*) **do**
  for each particle i in the swarm **do**
    /* **Update the position and the velocity** of the ith particle: */
    for $j = 1$ to D **do**
      $V^{i,j}_{g+1} = \chi\left(V^{i,j}_g + c_1 r_1\left(P^{i,j}_g - X^{i,j}_g\right) + c_2 r_2\left(P^{\text{best}_i,j}_g - X^{i,j}_g\right)\right)$,
      $X^{i,j}_{g+1} = X^{i,j}_g + V^{i,j}_{g+1}$.
    end for
    Calculate the fitness value, $f(X^i_{g+1})$, of the ith particle.
    /* **Update cognitive experience**: */
    if $f(X^i_{g+1}) < f(P^i_g)$
      $P^i_g = X^i_{g+1}$
    endif
  end for
  /* **Update social experience:** */
for each particle i in the swarm **do**
    for each neighbor X^n_{g+1} of the ith particle **do**
      if $f(P^i_{g+1}) < f(P^{\text{best}_i}_g)$ **then**
        $P^{\text{best}_i}_{g+1} = P^i_{g+1}$
      end if
    end for
  end for
  /* **Evolve social and cognitive experience:**/*{**Evolve** P^i_g **utilizing one DE step in** S^p_t */}
  for each P^i_g in the S^p_g **do**
    Select uniformly random integers $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\} \setminus \{i\}$
    /* **Mutate** P^i_g **and generate the mutant vector** v^i_g:*/
    for $j = 1$ to D **do**
      $v^{i,j}_g = P^{r_1,j}_g + F(P^{r_2,j}_g - P^{r_3,j}_g)$,
    end for
    /* **Recombine** the mutant vector v^i_g: */
    $j_{\text{rand}} =$ a uniformly distributed random integer $\in \{1, 2, \ldots, D\}$
    for $j = 1$ to D
      $$u^{i,j}_g = \begin{cases} v^{i,j}_g, & \text{if } (\text{rand}_{i,j}(0,1) \leqslant CR \text{or} j = j_{\text{rand}}), \\ P^{i,j}_g, & \text{otherwise}, \end{cases}.$$
    end for

(*continued on next page*)

Calculate the fitness value, $f(u_g^i)$ of the trial vector u_g^i.
/* **Update the** P_g^i through: */
$f(u_g^i) < f(P_g^i)$ **then**
$P_{g+1}^i = u_g^i$
else
$P_{g+1}^i = P_g^i$
endif
end for
end for

Focusing on the algorithmic design of the proposed framework, one can easily observe that, the proposed framework implements two main interacting modules on the *explorer-swarm*, as well as on the *memory-swarm*. PSO algorithm is executed with its classical structure, in an attempt to efficiently explore the search space and in parallel to focus on the most promising regions of the search space. Subsequently, DE operates only on the "good" experience that have been explored until the current step, i.e. the *memory-swarm*, aiming to effectively guide the algorithm towards the search of the global optimum. Notice that DE does not necessarily act as a "local search" procedure as in Memetic Algorithms [36,48], since it is a global search algorithm and has the potential not only to focus the search on particular regions, but also to explore unexplored regions of the search space. In detail, as illustrated in the Algorithm 1, the *explorer-swarm* of PSO is dedicated to the global search part of the framework, while the DE algorithm has a twofold role in the framework. By evolving the memory of the swarm, on the early stages of evolution DE helps the evolutionary process to potentially locate better unexplored regions of the problem at hand, while at the latter stages of evolution focuses the search on the most promising explored regions. It is evident that new hybrids with either a similar design or an entirely new one can be produced by giving emphasis in the algorithmic design modeling process, and use either bottom–up or top–down design methodologies, e.g. the algorithmic procedure proposed in [40].

Moreover, as illustrated in Algorithm 1, it should be noticed that the total computational cost of the proposed framework, in terms of function evaluations, includes the function evaluations of both PSO (PSO_{FEs}) and DE (DE_{FEs}) algorithms. Thus, the total amount of function evaluations equals to: $Total_{FE} = PSO_{FEs} + DE_{FEs}$. The only overhead of the resulting scheme is the operations of the applied algorithm to evolve the *memory-swarm* positions, i.e. in PSO:DE/rand/1/bin scheme the overhead depends only, (in terms of operations and not function evaluations), on the operations made by DE/rand/1/bin scheme on the personal best positions. As stated in the experimental results, for our implementation without any optimized source code, this overhead is on average at most 4% for the majority of the tested DE mutation strategies (please refer to Section 5.1.1).

4.2. Discussion on the hybridization of the PSO and the DE algorithms

The experimental results indicate that the proposed evolutionary hybrid framework assists both exploratory and exploitative behavior of the PSO algorithm. The evolution of the personal experience will initially promote the exploration of the personal experience space, since the personal experience is distributed almost randomly in the search space, while in the later time steps, where the personal bests have been gathered in the vicinity of local/global minima, will promote exploitation of the gathered experience. The proposed framework does not affect the main operations of PSO and can be directly applied to any PSO variant. According to our study, the proposed framework may result in great performance gains.

The hybrid PSO/DE algorithmic schemes can be classified using the notation PSO:DE/*base*/*num*/*cross*. PSO depicts to the PSO variant which will be used as the main PSO algorithm, while the DE/*base*/*num*/*cross* notation corresponds to the applied DE mutation strategy [27,17,83]. In detail, *base* indicates the base individual of the applied DE mutation strategy, *num* depicts the number of differences between individuals that are used to perturb the base individual, and *cross* stands for the crossover type utilized by the mutation strategy, i.e. *exp* for exponential and *bin* for binomial [27,17,83]. In this study, we always employ binomial crossover in the DE mutation strategies, and thus we exclude the *cross* part to simplify the notation. For example, the notation χPSO:DE/rand/1 represents the hybrid scheme that utilizes as the main PSO variant, the PSO with constriction factor (χPSO) and incorporates the DE/rand/1 mutation strategy to evolve its social and cognitive experience.

In general, the evolutionary process is a very efficient procedure, but it demands a high number of function evaluations to effectively converge to an optimum. The incorporation of an evolutionary algorithm in each evolution step may result in an increase of the required function evaluations. Thus, in cases where the function evaluations budget is limited, one can evolve only some best personal positions of the swarm in each generation. Several strategies can be applied to select which individuals should be evolved, such as evolve only the stagnated personal experience, or only the improved experience, or incorporate a probabilistically rule for the evolution. Some preliminary results have been presented in [25], where we have evolved only the best personal position that have been changed (improved) during the previous time step and may evolve to an even better position. Nevertheless, this strategy is not always the best one, since it depends on the evolution phase of the algorithm and the problem at hand. Thus, we propose to evolve all best personal positions at each time step, since the extensive experimental results presented in the next section, suggest that this procedure exhibits great performance gains.

Another notable observation is that it is not obligatory to evolve the personal experience set S_p^t using the Differential Evolution algorithm. Any efficient Evolutionary Algorithm can be applied. In initial experiments, we have utilized both local and global versions of the canonical PSO to evolve the S_p^t set. However, the results were not very promising and experiments on this direction were abandoned. It seems that utilizing methodologies with the same dynamics cannot provide significant performance gains. Such behavior has not been observed for the first time. For example, in [73], the authors have successfully evolved PSO control parameters using the DE algorithm, but when they tried to evolve the PSO parameters using the PSO algorithm the results were worse. This observation constitutes a very interesting research field for the evolutionary computing community, but it is not the main objective of the current work. We intent to study this phenomenon in a future work.

4.3. Discussion on rotation sensitivity and stagnation

In general hybridization approaches try to integrate several components, *building blocks*, within a general framework, in an attempt to combine their good characteristics and produce a more effective scheme. Nevertheless, we should always have in mind the deficiencies of each component and their impact on the resulting scheme. To this end, in this section we will briefly describe some aspects of the considered PSO and DE algorithms, that the interested reader should be aware of as well as how they might influence the proposed framework.

Firstly, it should be noticed that both PSO and DE algorithms suffer from two notable biases, they tend to perform very well when the optimum solution lies on or near the origin of the optimization search space (*origin biased*), as well as when the optimum solutions are either parallel to axis, or on the diagonal of the optimization search space (*rotation sensitive*). Thus, they operate better on separable, non-shifted and non-rotated landscapes with the optimum solution on, or near the origin of the optimization search space. On contrary, both DE and PSO encounter difficulties when handling non-separable, rotated landscapes or problems with shifted optima positions.

In the case of the PSO algorithm the responsible procedure is the position and velocity update rule that utilizes a dimension by dimension update [91,58]. An exposure of the origin-bias has been made in [58], in which two measures have been used to detect origin-seeking behavior; the region scaling and the center offset technique. Their experiments revealed that region scaling is an effective but not always sufficient test to detect this bias, while the center offset technique is more effective since it exposes cases that are not otherwise visible. Additionally, Spears et al. [58], showed that the rotational variance is related to the origin and axis bias of the PSO algorithm. Based on these observations they produced landscapes with specific characteristics that expose the performance ability of PSO to tackle them. In turn, an analysis on the impact of invariance in search of the PSO and the CMA-ES algorithms has been made in [34]. Through a thorough experimental investigation the authors argue that the incorporation of invariant characteristics, like rotational invariance, in an algorithmic scheme are desirable, because they increase the predictive power of performance results by inducing problem equivalence classes. The dimension by dimension update rule has been modified in the Standard PSO 2011 (SPSO 2011) [11], while other methodologies can been used to tackle the aforementioned biases, e.g. adaptive encoding [33].

In the case of the DE algorithm, the responsibility falls on the limited amount of DE search moves [83,94,41,42]. The vector–wise mutation scheme produces a limited amount of offspring, which strongly depends on the population size and the mutation factor value. In turn, the crossover operation essentially introduces sufficient diversity, but lacks the rotationally invariant property. Sutton et al. [94] observed that DE performs poorly on non-separable landscapes due to inefficient exploitation during the differential mutation phase. They explored two hypotheses regarding the crossover/recombination rate values: low values can exploit the separability of a landscape, while high values produce the desired rotationally invariant property which strongly depends on the differential mutation step. Thus, to efficiently handle non-separable landscapes, DE has to depend mostly on the mutation operation. In addition, setting the recombination rate equal to unity is not recommended, since it reduces the number of trial vectors which may result in *stagnation* [50]. The authors handled the aforementioned effects by imposing selective pressure through rank-based mutations [94]. Kenneth Price introduced in [82] a rotationally invariant DE algorithm, which eliminates drift bias from its trial vector generating function by projecting randomly chosen vector differences along lines of recombination. In addition, the aforementioned effects have been tackled in [41,42] by introducing the combinatorial sampling in DE, that provides a similar number of samples as crossover, without being biased towards the coordinate axes of the considered optimization search space. Thus, it increases the sampling diversity and is capable of tackling non-separable landscapes in high dimensional spaces even though the resulting scheme is not a strict rotationally invariant algorithm.

Additionally to the *rotation sensitivity*, another aspect that has to be taken into account is the risk of *stagnation* [50]. *Stagnation* is the undesired phase in which, as the optimization procedure progress and the population of the considered methodology remains diverse, the optimum seeking procedure stagnates before finding a globally optimum solution [50]. As mentioned previously, *stagnation* in DE occurs when it does not manage to improve upon any solution of its population for a extended number of evolution steps, i.e. produce a limited amount of exploratory moves. To improve the DE algorithm, several countermeasures have been considered against *stagnation*, with schemes/techniques that produce a randomization in its search operations to increase the amount of potential exploration moves and help the optimum seeking procedure to continue, e.g. the dither and jitter scale-factor schemes [83], multiple populations in parallel or distributed systems [97,109,103,63,104], sophisticated randomization schemes with adaptive of self-adaptive frameworks that efficiently change DE's control parameters [7,84,112,31,88], algorithmic adaptation of the scale-factor by means of local search [64,65], and scale-factor dynamics and inheritance in multiple populations [103,63,104] amongst others.

5. Experimental results

In this section we perform an extensive experimental evaluation of the proposed framework. We employ 25 high dimensional benchmark functions from the CEC 2005 benchmark suite on Real-Parameter Optimization [93]. More specifically, based on their characteristics, the CEC 2005 benchmark set functions can be divided into the following four classes: functions $f_1 - f_5$ are unimodal, $f_6 - f_{12}$ are basic multimodal functions, f_{13} and f_{14} are expanded multimodal functions, and $f_{15} - f_{25}$ are hybrid compositions of functions with a large number of local minima. A thorough description of this test set can be found in [93].

To provide a comprehensive comparison and highlight the different aspects of the proposed framework, we divide the presentation of the experimental results into the following subsections. We first incorporate the proposed framework into the classic version of PSO with constriction factor and demonstrate its main behavior (Section 5.1). Subsequently, we discuss the suitability of the proposed framework for five other well-known and widely used PSO variants (Section 5.2), and for four popular DE algorithms (Section 5.3). Finally, we conclude with an overall performance comparison among all the considered PSO variants (Section 5.4).

5.1. Hybridizing χPSO using the DE algorithm

In this section we hybridize the original χPSO algorithm with the Differential Evolution algorithm by incorporating six well-known DE mutation strategies with different characteristics. To maintain a reliable and fair comparison, we employ the same parameter settings for all PSO variants. Additionally, the swarms of all PSO variants, were initialized using a uniform random number distribution with the same random seeds.

In more detail, the parameter settings used are:

a. Swarm/population size: $NP = D$
b. PSO topology: *ring* with neighborhood radius $n_r = 2$
c. PSO parameters: $\phi = 4.1, \chi = 0.72984$, and $c_1 = c_2 = 2.05$ [6,12,77,74]
d. DE parameters: binomial crossover with $F = 0.5, CR = 0.9$ [92,8,55,7,27,17].

To evaluate the performance of the PSO variants we will use the *solution error measure*, or simply error, defined as $f(x') - f(x^\star)$, where $x^\star$ is the global optimum of the benchmark function and x' is the best solution achieved after $10^4 \cdot D$ function evaluations [93], where D is the dimensionality of the problem at hand. Each PSO variant was executed independently 100 times to obtain an estimation of the median (*Median*), the mean solution error (*Mean*), and its standard deviation (*St.D.*). For each pair of the original PSO variant and its corresponding hybrid DE-based variant, we use boldface font to indicate the best performance in terms of median solution error. To evaluate the statistical significance of the observed performance differences, we apply a two-sided Wilcoxon rank sum test between the original PSO variant and their hybrid DE-based variants. The null hypothesis in each test is that the samples compared are independent samples from identical continuous distributions with equal medians. When the null hypothesis is rejected at the 5% significance level, we mark with "+" the cases where the hybrid DE-based variant exhibits superior performance and with "−" when it exhibits inferior performance. When the performance difference is not statistically significant, we mark with "=". At the bottom of each table, for each pair, we also show the total number of the aforementioned statistical significant cases (+/=/−).

Based on the characteristics of the six DE mutation strategies (see Section 2.2), two groups can be distinguished. The first group includes mutation strategies with explorative dynamics [27], e.g. DE/rand/1, DE/rand/2, and TDE/rand/1, while the second group includes more exploitative mutation strategies, e.g. DE/best/1, DE/cur-to-best/1, and DE/best/2. To this end, Table 1 reports the results on the 30 and 50-dimensional versions of the CEC 2005 benchmark set for the explorative DE mutation strategies. One can clearly observe that the incorporation of the explorative DE mutation strategies in the χPSO algorithm yields either significant performance gains or operates similarly, with the χPSO:DE/rand/1 and χPSO:TDE/rand/1 variants to produce the best results.

More specifically, all hybrid PSO variants exhibit substantial performance improvements in all unimodal functions ($f_1 - f_5$), in almost all multimodal functions ($f_6, f_9 - f_{12}$), in the expanded function f_{13} and in the majority of the hybrid composition multimodal functions. On the other hand, the original χPSO algorithm operates better only on two multimodal functions f_7 and f_8, the expanded function f_{14} and on some 30-dimensional versions of the hybrid composition multimodal functions ($f_{18} - f_{20}, f_{22}$, and f_{25}). In the latter cases, it has to be noted that the three proposed hybrid PSO variants (χPSO:DE/rand/1, χPSO:DE/rand/2, and χPSO:TDE/rand/1) demonstrate significant performance improvements when the dimensionality increases, i.e. in the 50–dimensional versions of the aforementioned hybrid composition functions. Additionally, we can observe that in many benchmarks functions, the proposed hybrids with the explorative DE mutation strategies, provide the best results, between the χPSO and all the other hybrids. Specifically, χPSO:TDE/rand/1 performs best in 15, χPSO:DE/rand/1 in 10, and χPSO:DE/rand/1 in 9 out of 50 benchmark functions.

Table 2 depicts the results of the hybrid PSO variants with exploitative DE mutation strategies on the 30 and 50-dimensional versions of the CEC 2005 benchmark set. In these cases, the performance gains of the hybrids is less prominent. Generally speaking, based on the characteristics of the employed DE mutation strategy, we can observe that the more exploitative a mutation strategy is, the more it is performance deteriorates. More specifically, χPSO:DE/best/1 demonstrates

Table 1
Error values of the original local version of the χPSO algorithm and their corresponding explorative hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions.

f_i	D	χPSO			χPSO:DE/rand/1				χPSO:DE/rand/2				χPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	5.328e+00	9.657e+00	1.233e+01	**0.000e+00**	6.248e−02	2.184e−01	+	**0.000e+00**	5.226e−01	2.084e+00	+	**0.000e+00**	1.940e+00	7.555e+00	+
	50	0.000e+00	8.730e+01	1.950e+02	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
f_2	30	0.000e+00	1.573e+01	8.112e+01	**0.000e+00**	4.938e−02	2.745e−01	+	0.000e+00	1.088e−01	6.320e−01	=	0.000e+00	7.754e−02	4.013e−01	=
	50	2.334e+02	7.774e+02	1.806e+03	**2.000e−03**	3.040e−03	3.037e−03	+	**6.060e−01**	7.955e−01	8.022e−01	+	**1.100e−02**	5.807e+00	3.870e+01	+
f_3	30	3.491e+06	1.020e+07	1.336e+07	**6.060e+05**	8.767e+05	6.931e+05	+	**7.794e+05**	8.974e+05	6.196e+05	+	**7.484e+05**	8.967e+05	6.968e+05	+
	50	1.852e+07	1.988e+07	1.266e+07	**1.173e+06**	1.227e+06	4.163e+05	+	**4.949e+06**	5.680e+06	2.337e+06	+	**1.717e+06**	1.783e+06	7.190e+05	+
f_4	30	1.555e+03	1.834e+03	1.088e+03	**4.967e+01**	2.301e+02	5.235e+02	+	**1.876e+01**	1.496e+02	3.440e+02	+	**2.228e+01**	1.011e+02	2.212e+02	+
	50	2.813e+04	2.760e+04	1.007e+04	**1.066e+02**	6.108e+02	2.986e+03	+	**4.987e+02**	6.193e+02	5.710e+02	+	**4.273e+02**	7.573e+02	1.139e+03	+
f_5	30	7.886e+03	8.101e+03	1.210e+03	**4.754e+03**	4.818e+03	1.086e+03	+	**4.757e+03**	4.995e+03	1.185e+03	+	**4.792e+03**	4.932e+03	9.340e+02	+
	50	1.127e+04	1.117e+04	1.975e+03	**3.226e+03**	3.249e+03	7.024e+02	+	**2.785e+03**	2.782e+03	7.664e+02	+	**4.193e+03**	4.309e+03	9.041e+02	+
f_6	30	3.602e+02	1.170e+03	1.790e+03	**8.258e+01**	1.144e+04	6.378e+04	+	**1.351e+02**	6.497e+02	1.749e+03	+	**1.777e+02**	1.739e+03	7.807e+03	+
	50	3.979e+01	6.370e+06	2.129e+07	**3.759e+01**	6.101e+01	5.315e+01	=	**1.839e+01**	3.034e+01	3.701e+01	+	**3.965e+01**	5.108e+01	2.576e+01	=
f_7	30	**6.788e+03**	6.780e+03	1.291e+02	7.789e+03	7.774e+03	2.745e+02	−	7.940e+03	7.896e+03	2.389e+02	−	7.805e+03	7.759e+03	2.465e+02	−
	50	6.158e+03	6.154e+03	7.416e+01	**6.085e+03**	6.092e+03	8.772e+01	+	**6.080e+03**	6.075e+03	5.666e+01	+	**6.072e+03**	6.069e+03	6.803e+01	+
f_8	30	**2.090e+01**	2.090e+01	5.354e−02	2.093e+01	2.092e+01	5.683e−02	=	2.092e+01	2.092e+01	6.350e−02	=	2.093e+01	2.093e+01	5.436e−02	−
	50	**2.114e+01**	2.113e+01	4.368e−02	2.116e+01	2.115e+01	4.166e−02	−	2.116e+01	2.116e+01	3.032e−02	−	2.115e+01	2.115e+01	4.195e−02	−
f_9	30	6.517e+01	6.543e+01	1.239e+01	**4.875e+01**	5.034e+01	1.306e+01	+	**4.527e+01**	4.740e+01	1.184e+01	+	**4.726e+01**	4.762e+01	1.176e+01	+
	50	1.782e+02	1.765e+02	2.498e+01	**8.985e+01**	9.293e+01	2.582e+01	+	**1.062e+02**	1.056e+02	2.941e+01	+	**8.824e+01**	8.775e+01	2.017e+01	+
f_{10}	30	8.756e+01	8.665e+01	1.850e+01	**4.577e+01**	5.245e+01	2.654e+01	+	**4.975e+01**	5.765e+01	3.513e+01	+	**4.527e+01**	5.233e+01	3.107e+01	+
	50	1.837e+02	1.816e+02	3.686e+01	**8.434e+01**	8.907e+01	2.267e+01	+	**9.800e+01**	9.936e+01	2.329e+01	+	**8.094e+01**	8.271e+01	1.779e+01	+
f_{11}	30	2.802e+01	2.797e+01	2.487e+00	**2.206e+01**	2.252e+01	5.278e+00	+	**1.998e+01**	1.997e+01	4.524e+00	+	**1.930e+01**	2.016e+01	5.226e+00	+
	50	5.874e+01	5.817e+01	3.433e+00	**5.492e+01**	5.314e+01	8.229e+00	+	6.167e+01	6.121e+01	5.701e+00	−	**5.493e+01**	5.169e+01	9.416e+00	+
f_{12}	30	1.128e+04	1.872e+04	2.137e+04	**3.020e+03**	4.965e+03	5.692e+03	+	**4.754e+03**	5.489e+03	4.957e+03	+	**2.000e+03**	4.821e+03	7.071e+03	+
	50	2.896e+05	3.293e+05	1.922e+05	**1.070e+05**	1.294e+05	1.032e+05	+	**1.535e+05**	2.116e+05	1.669e+05	+	**7.944e+04**	1.152e+05	1.289e+05	+
f_{13}	30	5.389e+00	5.691e+00	1.775e+00	**3.355e+00**	3.324e+00	1.011e+00	+	**3.352e+00**	3.483e+00	1.055e+00	+	**3.168e+00**	3.127e+00	7.838e−01	+
	50	1.471e+01	1.543e+01	4.475e+00	**7.238e+00**	7.356e+00	1.734e+00	+	**7.797e+00**	7.913e+00	1.690e+00	+	**7.363e+00**	7.451e+00	1.923e+00	+
f_{14}	30	**1.211e+01**	1.205e+01	4.311e−01	1.222e+01	1.217e+01	4.283e−01	=	1.217e+01	1.219e+01	4.220e−01	=	1.237e+01	1.229e+01	4.841e−01	−
	50	2.193e+01	2.190e+01	4.937e−01	**2.190e+01**	2.188e+01	4.654e−01	=	2.211e+01	2.201e+01	5.579e−01	=	2.197e+01	2.194e+01	5.245e−01	=
f_{15}	30	4.036e+02	3.929e+02	8.708e+01	**4.006e+02**	3.825e+02	7.684e+01	=	**4.000e+02**	3.793e+02	7.077e+01	=	**4.023e+02**	3.981e+02	7.360e+01	=
	50	4.513e+02	4.198e+02	8.718e+01	**2.513e+02**	2.712e+02	7.002e+01	+	**2.553e+02**	2.785e+02	7.466e+01	+	**2.293e+02**	2.669e+02	8.084e+01	+
f_{16}	30	1.512e+02	1.867e+02	1.055e+02	**7.447e+01**	1.298e+02	1.120e+02	+	**7.839e+01**	1.669e+02	1.525e+02	+	**6.926e+01**	1.307e+02	1.246e+02	+
	50	1.679e+02	1.872e+02	5.268e+01	**8.700e+01**	1.051e+02	5.689e+01	+	**9.736e+01**	1.061e+02	3.337e+01	+	**7.893e+01**	9.170e+01	3.933e+01	+
f_{17}	30	1.881e+02	2.136e+02	8.914e+01	1.996e+02	2.074e+02	1.239e+02	=	**9.286e+01**	1.652e+02	1.267e+02	+	**1.867e+02**	1.930e+02	1.397e+02	=
	50	2.834e+02	2.890e+02	7.083e+01	**8.915e+01**	1.215e+02	7.626e+01	+	**1.078e+02**	1.371e+02	8.159e+01	+	**8.457e+01**	1.075e+02	7.727e+01	+
f_{18}	30	**9.746e+02**	9.388e+02	7.088e+01	9.796e+02	9.550e+02	7.632e+01	=	9.893e+02	9.827e+02	4.508e+01	−	9.802e+02	9.361e+02	8.747e+01	=
	50	9.376e+02	9.392e+02	8.816e+00	**9.174e+02**	9.193e+02	5.061e+00	+	**9.179e+02**	9.215e+02	8.786e+00	+	**9.215e+02**	9.225e+02	6.284e+00	+
f_{19}	30	**9.761e+02**	9.355e+02	6.813e+01	9.903e+02	9.658e+02	6.961e+01	−	9.766e+02	9.491e+02	8.101e+01	=	9.869e+02	9.621e+02	7.957e+01	−
	50	9.346e+02	9.383e+02	1.234e+01	**9.206e+02**	9.227e+02	9.796e+00	+	**9.186e+02**	9.207e+02	7.509e+00	+	**9.208e+02**	9.218e+02	6.888e+00	+
f_{20}	30	**9.756e+02**	9.553e+02	5.665e+01	9.921e+02	9.703e+02	6.754e+01	−	9.836e+02	9.469e+02	8.077e+01	=	9.958e+02	9.814e+02	4.994e+01	−
	50	9.371e+02	9.393e+02	1.197e+01	**9.211e+02**	9.231e+02	8.751e+00	+	**9.171e+02**	9.196e+02	6.372e+00	+	**9.207e+02**	9.205e+02	4.884e+00	+
f_{21}	30	5.098e+02	5.893e+02	2.082e+02	**5.001e+02**	6.357e+02	2.500e+02	+	**5.001e+02**	5.857e+02	2.048e+02	+	**5.001e+02**	5.984e+02	1.953e+02	+
	50	1.018e+03	1.019e+03	3.643e+00	**1.010e+03**	1.011e+03	3.544e+00	+	**1.012e+03**	1.013e+03	4.729e+00	+	**1.010e+03**	1.010e+03	3.471e+00	+
f_{22}	30	**1.024e+03**	1.026e+03	2.429e+01	1.029e+03	1.030e+03	3.078e+01	=	1.039e+03	1.035e+03	2.668e+01	=	1.029e+03	1.028e+03	3.003e+01	=
	50	9.095e+02	9.149e+02	2.018e+01	**8.982e+02**	9.024e+02	1.424e+01	+	**8.991e+02**	8.997e+02	1.053e+01	+	**8.930e+02**	8.976e+02	1.478e+01	+
f_{23}	30	5.068e+02	5.277e+02	1.018e+02	**5.003e+02**	5.743e+02	2.029e+02	+	**5.000e+02**	5.601e+02	1.689e+02	+	**5.002e+02**	5.986e+02	2.213e+02	+
	50	1.019e+03	1.019e+03	3.335e+00	**1.010e+03**	1.011e+03	3.540e+00	+	**1.011e+03**	1.011e+03	3.357e+00	+	**1.009e+03**	1.010e+03	4.263e+00	+

(*continued on next page*)

Table 1 (*continued*)

f_i	D	χPSO			χPSO:DE/rand/1				χPSO:DE/rand/2				χPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	2.000e+02	2.000e+02	0.000e+00	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	1.029e+03	1.018e+03	5.740e+01	**1.018e+03**	1.002e+03	1.158e+02	+	**1.024e+03**	1.024e+03	4.061e+00	+	**1.019e+03**	9.861e+02	1.621e+02	+
f_{25}	30	**1.750e+03**	1.750e+03	7.509e+00	1.763e+03	1.761e+03	1.350e+01	–	1.762e+03	1.760e+03	1.075e+01	–	1.764e+03	1.762e+03	1.169e+01	–
	50	1.682e+03	1.682e+03	5.316e+00	**1.671e+03**	1.671e+03	4.504e+00	+	**1.672e+03**	1.673e+03	5.690e+00	+	**1.671e+03**	1.671e+03	4.567e+00	+
	Total number of (+/–/=): (36/5/9)				Total number of (+/–/=): (36/5/9)				Total number of (+/–/=): (35/7/8)							

Table 2
Error values of the original local version of the χPSO algorithm and their corresponding exploitative hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions

f_i	D	χPSO			χPSO:DE/best/1				χPSO:DE/current-to-best/1				χPSO:DE/best/2			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	5.328e+00	9.657e+00	1.233e+01	**0.000e+00**	5.506e+00	2.754e+01	+	**0.000e+00**	8.049e+00	4.705e+01	+	**0.000e+00**	7.260e−01	2.968e+00	+
	50	**0.000e+00**	8.730e+01	1.950e+02	1.308e+03	2.029e+03	2.060e+03	−	0.000e+00	8.334e+01	3.453e+02	−	0.000e+00	2.657e+02	5.696e+02	=
f_2	30	0.000e+00	1.573e+01	8.112e+01	0.000e+00	2.880e−02	1.115e−01	=	0.000e+00	7.640e−03	3.400e−02	=	0.000e+00	2.830e−01	1.228e+00	=
	50	2.334e+02	7.774e+02	1.806e+03	7.983e+03	1.027e+04	8.913e+03	−	**1.101e+02**	1.116e+03	2.890e+03	+	2.736e+02	1.393e+03	2.228e+03	=
f_3	30	3.491e+06	1.020e+07	1.336e+07	**3.633e+05**	6.505e+05	6.267e+05	+	**6.205e+05**	8.280e+05	6.470e+05	+	**6.527e+05**	8.152e+05	6.276e+05	+
	50	1.852e+07	1.988e+07	1.266e+07	**9.060e+06**	1.780e+07	2.068e+07	+	**2.388e+06**	4.404e+06	5.730e+06	+	**4.881e+06**	7.594e+06	8.338e+06	+
f_4	30	1.555e+03	1.834e+03	1.088e+03	**3.261e+01**	1.974e+02	4.063e+02	+	**1.352e+01**	1.170e+02	2.025e+02	+	**4.239e+00**	1.430e+02	3.571e+02	+
	50	2.813e+04	2.760e+04	1.007e+04	**2.189e+04**	2.284e+04	1.033e+04	+	**9.155e+03**	9.733e+03	4.889e+03	+	**6.808e+02**	2.724e+03	5.027e+03	+
f_5	30	7.886e+03	8.101e+03	1.210e+03	**4.425e+03**	4.610e+03	1.368e+03	+	**4.656e+03**	4.787e+03	1.118e+03	+	**5.023e+03**	4.958e+03	1.180e+03	+
	50	1.127e+04	1.117e+04	1.975e+03	**1.115e+04**	1.082e+04	2.528e+03	=	**1.102e+04**	1.093e+04	2.220e+03	=	**5.427e+03**	5.629e+03	1.761e+03	+
f_6	30	3.602e+02	1.170e+03	1.790e+03	**1.331e+02**	2.586e+04	1.337e+05	+	**8.769e+01**	2.626e+02	6.543e+02	+	**7.841e+01**	9.781e+03	6.693e+04	+
	50	**3.979e+01**	6.370e+06	2.129e+07	2.796e+07	3.507e+08	9.003e+08	−	8.359e+01	3.065e+07	1.246e+08	=	2.111e+02	2.022e+07	7.655e+07	−
f_7	30	**6.788e+03**	6.780e+03	1.291e+02	7.860e+03	7.815e+03	2.495e+02	−	7.841e+03	7.835e+03	2.356e+02	−	7.853e+03	7.801e+03	2.926e+02	−
	50	6.158e+03	6.154e+03	7.416e+01	**6.150e+03**	6.158e+03	1.550e+02	=	**6.140e+03**	6.123e+03	8.054e+01	=	**6.062e+03**	6.071e+03	1.173e+02	+
f_8	30	**2.090e+01**	2.090e+01	5.354e−02	2.094e+01	2.092e+01	6.115e−02	=	2.093e+01	2.093e+01	5.664e−02	−	2.093e+01	2.092e+01	5.522e−02	=
	50	2.114e+01	2.113e+01	4.368e−02	2.116e+01	2.115e+01	3.641e−02	−	**2.114e+01**	2.113e+01	4.199e−02	=	2.116e+01	2.115e+01	3.738e−02	−
f_9	30	6.517e+01	6.543e+01	1.239e+01	**5.074e+01**	4.949e+01	1.373e+01	+	**4.875e+01**	4.975e+01	1.195e+01	+	**4.477e+01**	4.704e+01	1.074e+01	+
	50	1.782e+02	1.765e+02	2.498e+01	**1.697e+02**	1.771e+02	3.791e+01	=	**1.210e+02**	1.193e+02	2.348e+01	+	**1.258e+02**	1.230e+02	2.747e+01	+
f_{10}	30	8.756e+01	8.665e+01	1.850e+01	**4.577e+01**	5.536e+01	3.384e+01	+	**5.174e+01**	5.417e+01	1.838e+01	+	**4.676e+01**	5.486e+01	3.575e+01	+
	50	1.837e+02	1.816e+02	3.686e+01	1.847e+02	1.890e+02	4.366e+01	=	**1.451e+02**	1.409e+02	3.244e+01	+	**1.175e+02**	1.193e+02	2.272e+01	+
f_{11}	30	2.802e+01	2.797e+01	2.487e+00	**1.929e+01**	2.064e+01	5.435e+00	+	**2.166e+01**	2.209e+01	5.396e+00	+	**2.253e+01**	2.223e+01	4.757e+00	+
	50	5.874e+01	5.817e+01	3.433e+00	**4.889e+01**	5.099e+01	6.702e+00	+	**5.456e+01**	5.341e+01	5.885e+00	+	**5.266e+01**	5.211e+01	7.780e+00	+
f_{12}	30	1.128e+04	1.872e+04	2.137e+04	**3.362e+03**	4.627e+03	4.789e+03	+	**2.404e+03**	3.721e+03	3.491e+03	+	**3.277e+03**	5.759e+03	7.156e+03	+
	50	2.896e+05	3.293e+05	1.922e+05	3.449e+05	4.970e+05	5.155e+05	=	**1.101e+05**	1.313e+05	1.101e+05	+	**1.887e+05**	2.154e+05	1.403e+05	+
f_{13}	30	5.389e+00	5.691e+00	1.775e+00	**3.120e+00**	3.158e+00	8.580e−01	+	**3.244e+00**	3.302e+00	8.529e−01	+	**3.242e+00**	3.313e+00	9.144e−01	+
	50	1.471e+01	1.543e+01	4.475e+00	2.499e+01	2.642e+01	9.929e+00	−	1.794e+01	2.002e+01	6.408e+00	−	**1.025e+01**	1.029e+01	2.643e+00	+
f_{14}	30	**1.211e+01**	1.205e+01	4.311e−01	1.226e+01	1.220e+01	4.412e−01	=	1.222e+01	1.215e+01	4.916e−01	=	1.233e+01	1.226e+01	4.293e−01	−
	50	2.193e+01	2.190e+01	4.937e−01	**2.127e+01**	2.111e+01	7.217e−01	+	**2.138e+01**	2.135e+01	5.359e−01	+	**2.189e+01**	2.185e+01	5.111e−01	=
f_{15}	30	4.036e+02	3.929e+02	8.708e+01	**4.000e+02**	3.827e+02	6.628e+01	=	**4.015e+02**	3.821e+02	9.402e+01	=	**4.010e+02**	3.911e+02	6.197e+01	=
	50	4.513e+02	4.198e+02	8.718e+01	5.540e+02	5.519e+02	7.497e+01	−	**4.467e+02**	4.519e+02	5.049e+01	=	**4.378e+02**	4.305e+02	6.778e+01	=
f_{16}	30	1.512e+02	1.867e+02	1.055e+02	**7.383e+01**	1.337e+02	1.304e+02	+	**7.875e+01**	1.642e+02	1.584e+02	+	**7.219e+01**	1.334e+02	1.217e+02	+
	50	1.679e+02	1.872e+02	5.268e+01	1.844e+02	2.283e+02	1.166e+02	=	**1.286e+02**	1.708e+02	1.124e+02	+	**1.277e+02**	1.806e+02	1.191e+02	+
f_{17}	30	1.881e+02	2.136e+02	8.914e+01	1.982e+02	2.017e+02	1.403e+02	=	1.899e+02	2.022e+02	1.484e+02	=	**1.489e+02**	1.762e+02	1.285e+02	+
	50	2.834e+02	2.890e+02	7.083e+01	**1.944e+02**	2.492e+02	1.261e+02	+	**1.295e+02**	1.667e+02	1.042e+02	+	**1.127e+02**	1.551e+02	1.104e+02	+
f_{18}	30	**9.746e+02**	9.388e+02	7.088e+01	9.847e+02	9.613e+02	6.849e+01	−	9.855e+02	9.588e+02	7.691e+01	−	9.862e+02	9.555e+02	7.531e+01	−
	50	**9.376e+02**	9.392e+02	8.816e+00	9.645e+02	9.696e+02	2.955e+01	−	9.556e+02	9.575e+02	2.042e+01	−	9.397e+02	9.426e+02	1.887e+01	=
f_{19}	30	**9.761e+02**	9.355e+02	6.813e+01	9.950e+02	9.517e+02	8.768e+01	−	9.805e+02	9.393e+02	8.442e+01	=	9.774e+02	9.481e+02	8.170e+01	=
	50	**9.346e+02**	9.383e+02	1.234e+01	9.690e+02	9.723e+02	2.407e+01	−	9.537e+02	9.600e+02	2.254e+01	−	9.373e+02	9.383e+02	1.320e+01	=
f_{20}	30	**9.756e+02**	9.553e+02	5.665e+01	9.831e+02	9.355e+02	8.549e+01	=	9.839e+02	9.707e+02	5.447e+01	=	9.894e+02	9.693e+02	6.037e+01	−
	50	9.371e+02	9.393e+02	1.197e+01	9.697e+02	9.737e+02	2.910e+01	−	9.540e+02	9.617e+02	2.803e+01	−	**9.299e+02**	9.343e+02	1.247e+01	+
f_{21}	30	5.098e+02	5.893e+02	2.082e+02	**5.000e+02**	5.989e+02	2.303e+02	+	**5.000e+02**	5.472e+02	1.610e+02	+	**5.001e+02**	6.059e+02	2.324e+02	+
	50	1.018e+03	1.019e+03	3.643e+00	1.025e+03	1.018e+03	8.232e+01	−	**1.017e+03**	9.813e+02	1.549e+02	=	**1.016e+03**	1.018e+03	7.491e+00	+
f_{22}	30	1.024e+03	1.026e+03	2.429e+01	1.027e+03	1.031e+03	3.079e+01	=	1.030e+03	1.034e+03	3.300e+01	=	**1.023e+03**	1.031e+03	2.738e+01	=
	50	**9.095e+02**	9.149e+02	2.018e+01	9.495e+02	9.523e+02	4.360e+01	−	9.406e+02	9.419e+02	3.157e+01	−	9.127e+02	9.227e+02	3.087e+01	=
f_{23}	30	5.068e+02	5.277e+02	1.018e+02	**5.000e+02**	6.371e+02	2.553e+02	+	**5.001e+02**	5.912e+02	2.183e+02	+	**5.006e+02**	6.124e+02	2.429e+02	+
	50	1.019e+03	1.019e+03	3.335e+00	1.025e+03	1.029e+03	6.805e+01	−	**1.018e+03**	9.967e+02	1.413e+02	=	**1.016e+03**	1.017e+03	5.825e+00	+

(*continued on next page*)

Table 2 (*continued*)

f_i	D	χPSO			χPSO:DE/best/1				χPSO:DE/current-to-best/1				χPSO:DE/best/2			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	2.000e+02	2.000e+02	0.000e+00	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.060e+02	4.243e+01	=	2.000e+02	2.000e+02	0.000e+00	=
	50	1.029e+03	1.018e+03	5.740e+01	**1.026e+03**	1.042e+03	5.751e+01	+	**1.019e+03**	8.878e+02	2.391e+02	+	**1.020e+03**	1.022e+03	8.851e+00	+
f_{25}	30	**1.750e+03**	1.750e+03	7.509e+00	1.760e+03	1.760e+03	1.187e+01	–	1.762e+03	1.762e+03	1.037e+01	–	1.760e+03	1.761e+03	1.070e+01	–
	50	1.682e+03	1.682e+03	5.316e+00	1.686e+03	1.689e+03	1.273e+01	–	**1.675e+03**	1.677e+03	7.781e+00	+	**1.677e+03**	1.678e+03	8.306e+00	+
					Total number of (+/–/=): (19/17/14)				Total number of (+/–/=): (25/10/15)				Total number of (+/–/=): (30/7/13)			

the worst performance between all hybrid PSO variants, since it deteriorates the performance of the original χPSO algorithm in 17 out of 50 cases. χPSO:DE/best/1 results in significant improvements only in three unimodal functions ($f_3 - f_5$), in four multimodal ($f_9 - f_{12}$), and in some specific cases of the hybrid composition functions, i.e. in the 30-dimensional cases of f_{16}, f_{21}, and f_{23}, and the 50-dimensional cases of f_{17} and f_{24}. Therefore, χPSO:DE/cur-to-best/1 demonstrates significant performance difference in comparison to the original χPSO algorithm in three unimodal functions ($f_2 - f_5$), in four multimodal ($f_9 - f_{12}$), and in some hybrid composition functions. This behavior, totally changes in the case of χPSO:DE/best/2 hybrid. The incorporation of a second difference vector in the DE/best/2 mutation strategy, enhances the exploratory power of the mutation procedure, which significantly improves the performance of the χPSO algorithm. Thereby χPSO:DE/best/2 is statistically better than χPSO in 35 out of 50 benchmark functions. It has a great impact on almost all unimodal functions ($f_1 - f_5$), three multimodal and one expanded function ($f_9 - f_{13}$), as well as on several hybrid composition functions ($f_{15} - f_{17}$, and $f_{21} - f_{24}$).

In previous works, we have observed that DE mutation strategies exhibit different behavior based on their explorative/exploitative dynamics. Exploitative strategies rapidly gather all the individuals to the basin of attraction of a single minimum, whereas explorative strategies tend to spread the individuals around many minima [27,98,23]. Thereby, the aforementioned behavior of the hybrid PSO variants was expected. The application of an exploitative mutation strategy decreases the power of the cognitive and social experience, resulting in a more exploitative scheme, that generally performs worse than the original PSO variant. On the other hand, as it is validated by the aforementioned experimental results, for the majority of the considered benchmark functions, the incorporation of an explorative DE mutation strategy yields hybrid PSO variants with significant performance gains. Thus, its incorporation to a PSO variant is highly recommended.

Furthermore, it is very interesting to evaluate the impact of the proposed framework in a global PSO variant, since the global version of a PSO algorithm utilizes a fully connected topology, which leads to a very exploitative optimization algorithm. Recall that the global version of a PSO algorithm includes the global best particle in its velocity update rule.

Initially, one can suppose that the incorporation of the proposed framework, especially with an exploitative DE mutation strategy, may result in a very inefficient PSO variant. Nevertheless, we have utilized the proposed hybridization framework in the global version of the χPSO algorithm. The hybrid PSO variants with either explorative or exploitative DE mutation strategies, exhibit either significant performance differences or operates similarly for the majority of the benchmark functions. Specifically, the performance of the global χPSO variant has been significantly enhanced in all unimodal ($f_1 - f_5$), expanded (f_{13} and f_{14}), and hybrid composition functions ($f_{15} - f_{25}$) and in the majority of the multimodal functions. The performance of the global χPSO variant has been deteriorated by the proposed framework, only in f_7 and f_8 functions. Therefore, the hybridization of the global PSO version with any DE mutation strategy is also highly recommended. Due to lack of space the corresponding tables have been excluded from this paper, but they will be provided to the interested reader upon request.

5.1.1. CPU time overhead of the proposed framework

To demonstrate the computational time overhead of the proposed framework, we calculate the wall clock CPU time of the PSO algorithm and the corresponding hybrid variants on each benchmark function. Thereby, the computational time overhead of a hybrid against a PSO algorithm on the benchmark function f, Overhead(f), can been calculated according to the following formula:

$$\text{Overhead}(f) = \frac{\text{PSO} : \text{DE}_{\text{time}}(f) - \text{PSO}_{\text{time}}(f)}{\text{PSO}_{\text{time}}(f)}, \tag{12}$$

where $\text{PSO}_{\text{time}}(f)$ and $\text{PSO} : \text{DE}_{\text{time}}(f)$, indicate the mean value of the wall clock CPU time measured, on the benchmark function f over the conducted simulations, for the PSO and the PSO:DE hybrid algorithm respectively. As such, Fig. 5 illustrates the computational time overhead of the proposed hybrid DE strategies against the χPSO algorithm on the 30-dimensional versions (top) and the 50-dimensional versions (bottom) of the CEC 2005 suite. It can be clearly observed that on average the computational time overhead is below 2% and 4% for the majority of the proposed hybrids, over the 30-dimensional and 50-dimensional versions of the considered function suite, respectively. Only DE/best/2 exhibits an overhead between 3% and 5% for some 30-dimensional benchmark functions. Let us remind you that the implementation of the proposed framework does not contain any source code optimization, thus the computational time overhead could be further reduced.

5.1.2. Statistical significance analysis of the hybrid χPSO Variants

To evaluate the statistical significance of the observed performance differences, we additionally provide an extensive statistical analysis [18,30]. The analysis firstly includes statistical test procedures that can rank the performance of a set of k algorithms and answer whether there is a significant deference in the performance of at least two of those algorithms. If there is a significant difference, we can continue by applying a post hoc test analysis to determine in which cases the best performing algorithm exhibits a significant variation. The statistical analysis ends with the application of the Empirical Cumulative probability Distribution Function graph of the performance error (ECDF). ECDF graph demonstrates the performance of all algorithms in all test cases and can be utilized as an overall performance visualization statistic.

More specifically, we firstly utilize three procedures for testing the differences between more than two related samples, namely, the Friedman, the Aligned Friedman, and the Quade tests [13,18,30,89]. The Friedman test (Friedman two-way analysis of variances by ranks) is a non-parametric multiple comparisons test, which aims to detect significant differences

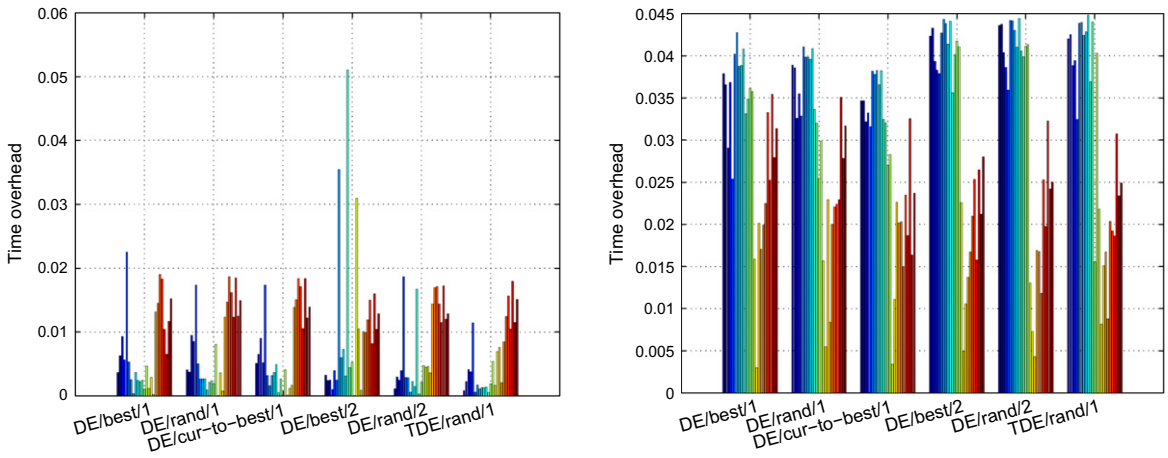

Fig. 5. Computational time overhead of the proposed hybrid DE strategies against the χPSO algorithm on the 30-dimensional versions (top) and the 50-dimensional versions (bottom) of the CEC 2005 suite.

between the behavior of two or more population samples. It answers if in a set of k population samples, with $k > 2$, there are at least two population samples with different median values. The null hypothesis states that all population samples have equal medians, while the alternative hypothesis is defined as the negation of the null hypothesis. The Aligned Friedman test is a variation of the Friedman test, which utilizes aligned ranks to handle intra-block effects in a population sample [13,30,89]. Finally, the Quade test is a variation of the Friedman test, which takes into account the fact that some cases in the population sample may be more important than others (Friedman test considers all cases to be equal in terms of importance) [30]. Thereby, the rankings computed by the Quade test could be scaled depending on the differences observed in the samples. Consequently, if there is a significant difference, we can continue by applying a post hoc test analysis to determine in which cases the best performing algorithm (*control method*) exhibits a significant variation. Here, we utilize post hoc tests with different abilities and characteristics [13,18,30,89].

Finally, to provide a summarizing comparison of the implemented algorithms on all benchmark functions, we utilize the Empirical Cumulative probability Distribution Function (*ECDF*) of the performance error values. Specifically, for an algorithm A on a function f, the error value (error) achieved by A on function f is computed. Therefore, smaller values of error correspond to better performance. The *ECDF* of errors for an algorithm in a number of functions n_f is a cumulative probability distribution function defined as:

$$ECDF(x) = \frac{1}{n_f}\sum_{i=1}^{n_f} I(\text{error}_i \leqslant x),$$

where $I(\cdot)$ is the indicator function. In other words, the *ECDF* measure captures the empirical probability of observing an error value smaller or equal to x. First, we compute the errors for all considered algorithms on all the functions and then we compute the *ECDF* for each algorithm. This enables a summarizing comparison of the algorithms in all the benchmarks, as larger *ECDF* values for the same argument correspond to better performance.

The statistical analysis starts with Table 3, which depicts the average rankings computed through the Friedman, the Aligned Friedman, and the Quade tests. At the bottom of the table we demonstrate the statistics of each test along with its corresponding p-values. The p-values computed through the statistics of the first two statistical tests ($9.291e-7$, $2.541e-7$) and the Iman and Davenport extension ($F_f = 7.1972$, p-value: $3.531e-7$), strongly suggests the existence of significant differences among the considered algorithms, at the level of significance $\alpha = 0.05$. The p-value of the Quade test rejects the null hypothesis at the significance level of $\alpha = 0.05$, while accepts it at $\alpha = 0.1$. Thus, the Quade test suggests that the corresponding algorithms does not exhibit significant differences in the most difficult benchmark functions at the level of significance $\alpha = 0.05$, while there are significant differences if we increase the level at $\alpha = 0.1$. Additionally, Table 3 highlights the ranking of the considered algorithms. It is obvious to observe that the original χPSO algorithm always comes in the last position of the rankings, while the first two positions belong to the χPSO:DE/rand/1 and the χPSO:TDE/rand/1 hybrids. As expected by the aforementioned extensive results, the first three positions always belong to the hybrid PSO variants with explorative mutation strategies, while the next three positions always belong to the hybrids with exploitative mutation strategies.

To detect the cases in which the best performing algorithm (*control method*) exhibits a significant variation, we continue with the post hoc analysis. Table 4 demonstrates the results of the post hoc tests for the Friedman, Aligned Friedman, and Quade tests. For each test, we exhibit the p-values of the post hoc tests at the level of significance $\alpha = 0.05$. Notice that the control method of the Friedman and the Quade test is the χPSO:TDE/rand/1 hybrid, while for the Aligned Friedman test is the χPSO:DE/rand/1 hybrid. We can safely conclude that for each test, the corresponding control method is always significantly

Table 3
Average Rankings of the local version of the χPSO variants, achieved by the Friedman, Aligned Friedman, and Quade statistical tests.

Algorithms	Average ranking		
	Friedman	Aligned Friedman	Quade
χPSO	5.150 (7)	243.470 (7)	4.809 (7)
χPSO:DE/best/1	4.810 (6)	234.129 (6)	4.326 (6)
χPSO:DE/rand/1	3.220 (2)	132.739 (1)	3.632 (2)
χPSO:DE/cur-to-best/1	4.190 (5)	187.489 (5)	4.141 (5)
χPSO:DE/best/2	3.940 (4)	159.280 (4)	3.957 (4)
χPSO:DE/rand/2	3.570 (3)	135.710 (3)	3.637 (3)
χPSO:TDE/rand/1	3.120 (1)	135.680 (2)	3.495 (1)
Statistic	38.4214	41.2879	1.7939
p-value	9.291e−7	2.541e−7	0.100

Table 4
Post-hoc tests analysis for the Friedman, Aligned Friedman, and Quade tests. For each test, we provide the *p*-values of the Holm, Rom, and Finner post hoc tests at the level of significance $\alpha = 0.05$.

Algorithm	*z*-Value	*p*-Value	Holm	Rom	Finner
Post-hoc p-values for the Friedman test, with control method: χPSO:TDE/rand/1					
χPSO	4.6985	2.6203e−6	0.0083 (+)	0.0087 (+)	0.0085 (+)
χPSO:DE/best/1	3.9115	9.1690e−5	0.0100 (+)	0.0105 (+)	0.0169 (+)
χPSO:DE/cur-to-best/1	2.4765	0.0132	0.0125 (+)	0.0131 (=)	0.0253 (+)
χPSO:DE/best/2	1.8979	0.0577	0.0166 (=)	0.0166 (=)	0.0336 (+)
χPSO:DE/rand/2	1.0415	0.2976	0.0250 (=)	0.0250 (=)	0.0418 (=)
χPSO:DE/rand/1	0.2314	0.8169	0.0500 (=)	0.0500 (=)	0.0500 (=)
Post-hoc p-values for the Aligned Friedman test, with control method: χPSO:DE/rand/1					
χPSO	5.4719	4.4522e−8	0.0083 (+)	0.0087 (+)	0.0085 (+)
χPSO:DE/best/1	5.0103	5.4330e−7	0.0100 (+)	0.0105 (+)	0.0169 (+)
χPSO:DE/cur-to-best/1	2.7055	0.0068	0.0125 (+)	0.0131 (+)	0.0253 (+)
χPSO:DE/best/2	1.3115	0.1896	0.0166 (+)	0.0166 (=)	0.0336 (+)
χPSO:DE/rand/2	0.1467	0.8833	0.0250 (=)	0.0250 (=)	0.0418 (=)
χPSO:TDE/rand/1	0.1452	0.8844	0.0500 (=)	0.0500 (=)	0.0500 (=)
Post-hoc p-values for the Quade test, with control method: χPSO:TDE/rand/1					
χPSO	1.5287	0.1263	0.0083 (+)	0.0087 (+)	0.0085 (+)
χPSO:DE/best/1	0.9668	0.3336	0.0100 (=)	0.0105 (=)	0.0169 (=)
χPSO:DE/cur-to-best/1	0.7511	0.4525	0.0125 (=)	0.0131 (=)	0.0253 (=)
χPSO:DE/best/2	0.5372	0.5910	0.0166 (=)	0.0166 (=)	0.0336 (=)
χPSO:DE/rand/2	0.1655	0.8685	0.0250 (=)	0.0250 (=)	0.0418 (=)
χPSO:DE/rand/1	0.1591	0.8735	0.0500 (=)	0.0500 (=)	0.0500 (=)

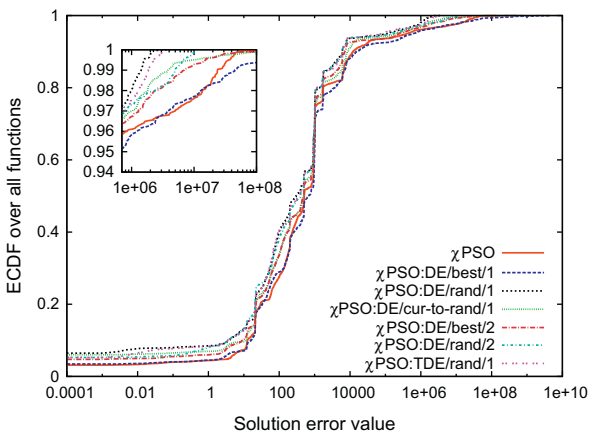

Fig. 6. Empirical cumulative probability distribution of the solution error values for the local version of the χPSO algorithm against the corresponding hybrid DE variants on the 30 and 50-dimensional versions of the CEC 2005 benchmark functions.

different to the original χPSO algorithm. Additionally, from the first two tests, we can highlight that there is a significant difference between the corresponding control method and both χPSO:DE/best/1 and χPSO:DE/cur-to-best/1 hybrids.

Finally, to provide a summarizing comparison of the proposed framework on all benchmark functions, we utilize the Empirical Cumulative probability Distribution Function (*ECDF*) of the solution error values on all 30 and 50-dimensional versions of the CEC 2005 benchmark functions. Fig. 6, illustrates the *ECDF* of the error for the original local version of χPSO versus its hybrid DE variants. As it can be clearly observed, the hybrid DE variants of the χPSO algorithm exhibit a great potential on the CEC 2005 function set. Almost all hybrid DE variants exhibit higher ECDF values compared with the original χPSO algorithm, except for the χPSO:DE:best/1, which demonstrates lower ECDF values in many solution error values. In general, the χPSO:DE:rand/1 and χPSO:TDE:rand/1 produce one to two orders of magnitude less error values than the original χPSO algorithm, i.e. the χPSO:DE:rand/1 and χPSO:TDE:rand/1 curves reach unity at approximate error10^6, while the original χPSO curve at approximate error10^8 (please refer to the zoomed subfigure inside Fig. 6). A similar behavior can be observed for the χPSO:DE:rand/2 variant, where it reaches unity at approximate error10^7.

5.2. Hybridization of state-of-the-art PSO variants

In this subsection, we apply the hybrid DE-based framework on five well known and widely used PSO variants. Based on the aforementioned analysis on the χPSO algorithm, we apply to the PSO variants the three best performing DE mutation strategies i.e. the explorative DE/rand/1, DE/rand/2, and TDE/rand/1. Specifically, we incorporate the proposed framework on: (i) the Bare Bones Particle Swarm Optimization (BBPSO) [46], (ii) the Dynamic Multi Swarm Particle Swarm Optimization (DMSPSO) [53], (iii) the Fully Informed Particle Swarm Optimization (FIPS) [57], (iv) the Unified Particle Swarm Optimization (UPSO) [75,76], and (v) the Comprehensive Learning Particle Swarm Optimizer (CLPSO) [52].

We evaluate the performance of the five PSO variants and their corresponding DE-based modifications on the 30 and the 50-dimensional versions of the CEC 2005 function set. Tables 5–9 report their comprehensive experimental results. It can be easily observed that the hybrid approaches exhibit similarities on their performance, so based on their performance similarities, we can distinguish them on three groups; BBPSO and DMSPSO, FIPS and UPSO, and CLPSO.

For the first group, we can easily observe that all hybrid variants result either significant performance gains or operate similarly, on all unimodal and the majority of hybrid composition functions. In the multimodal function cases, each algorithm exhibits different performance, but in most of the cases the hybrids outperform the corresponding PSO algorithms.

More specifically, in the BBPSO case, the proposed hybrids substantially influence its performance. Table 5, highlights that all hybrid variants exhibit significant improvements on all unimodal ($f_1 - f_5$), the majority of hybrid composition ($f_{18} - f_{25}$), and on five multimodal functions ($f_6 - f_9$, and f_{12}). On the contrary, the performance of the BBPSO algorithm is deteriorated by the hybrid DE mutation strategies only on 6 functions, i.e. in two multimodal (f_{10} and f_{11}), one expanded f_{14}, and three hybrid composition functions ($f_{15} - f_{17}$). Concluding, we observe that between the three hybrids the best performing variant, in terms of lower error values, is the BBPSO:DE/rand/1; second comes BBPSO:TDE/rand/1, while next comes the BBPSO:DE/rand/2 hybrid. This observation is validated through the aforementioned statistical ranking tests and the corresponding post hoc analysis, which strongly suggest that there exist statistical significant performance differences among the considered hybrids and the BBPSO algorithm. Trying to rationalize the highlighted improvements, we recall that BBPSO updates its particles positions though a Gaussian distribution, which is based on the particle's best personal (cognitive) and the best neighborhood (social) positions. Thus, as the experimental results suggest, an intelligent evolution of the aforementioned positions can lead them to more promising regions of the optimization space and enhance the power of the utilized distribution, resulting in a scheme with substantial performance gains.

A similar behavior is observed in the case of the DMSPSO algorithm, as demonstrated in Table 6. Let us remind you that DMSPSO implements multiple small swarms that randomly regroup, in an attempt to introduce a dynamically changing neighborhood structure to each particle. It can be clearly observed in the reported results that the hybridization framework does not affect the design of DMSPSO; on the contrary, the hybrid algorithms exhibit significant performance gains. There are only 7 out of 50 cases where the proposed framework exhibits deteriorated performance against DMSPSO; the multimodal f_8 function, the expanded f_{14} function, and the f_{10}, f_{11} and f_{17} 50–dimensional cases. Here, the best performing algorithm is the DMSPSO:DE/rand/1, since it produces significantly superior performance in 40 out of 50 benchmark functions. The DMSPSO:TDE/rand/1 variant follows, which demonstrates significant performance enhancements on 36 benchmark functions. Although DMSPSO:DE/rand/2 produces significant improvements in most of the unimodal (f_1, f_4, and f_5) and the hybrid composition functions ($f_{17} - f_{20}$ and f_{24}), it hinders the performance of DMSPSO in four multimodal functions (f_8, f_{10}, f_{11}, and f_{14}). The aforementioned observations have been validated by statistical ranking tests, which indicate that there are statistical significant performance differences between the original DMSPSO and all the hybrid algorithm. It has been also observed that DMSPSO:DE/rand/1 algorithm always exhibits significant performance differences against the DMSPSO:DE/rand/2. Finally, the performance difference between DMSPSO:DE/rand/1 and DMSPSO:TDE/rand/1 is not statistical significant.

The second group consists of the Fully Informed PSO (FIPS) [57] and the Unified PSO (UPSO) [75,76] algorithms. Recall that FIPS depends on the particle's best personal positions, since it produces new particles by calculating the centroid of its neighboring best personal positions. Additionally, UPSO algorithm utilizes a velocity update scheme that efficiently combines the local and global versions of the χPSO algorithm to balance their explorative and exploitative characteristics. For each particle, it incorporates four best personal positions in its velocity update rule, i.e. the global best, the best neighborhood, and two times the best personal position of the corresponding particle. Consequently, their evolution and

Table 5
Error values of the BBPSO algorithm and their corresponding explorative hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions.

f_i	D	BBPSO			BBPSO:DE/rand/1				BBPSO:DE/rand/2				BBPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=
	50	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=
f_2	30	6.000e−03	9.260e−03	8.480e−03	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	7.200e−04	2.295e−03	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	2.407e+02	2.886e+02	1.452e+02	**6.600e−02**	9.474e−02	9.389e−02	+	6.257e+03	6.169e+03	2.161e+03	−	**4.234e+01**	5.274e+01	4.550e+01	+
f_3	30	1.243e+06	1.295e+06	5.728e+05	**2.884e+05**	3.398e+05	1.905e+05	+	**6.326e+05**	6.659e+05	3.727e+05	+	**4.366e+05**	4.476e+05	1.635e+05	+
	50	3.693e+06	3.709e+06	9.352e+05	**8.047e+05**	8.501e+05	2.933e+05	+	2.445e+07	2.565e+07	1.008e+07	+	**1.202e+06**	1.365e+06	5.064e+05	+
f_4	30	1.962e+03	2.307e+03	1.188e+03	**4.600e−02**	4.047e−01	1.048e+00	+	**1.436e+00**	2.362e+00	4.151e+00	+	**5.250e−02**	3.123e−01	1.149e+00	+
	50	3.026e+04	2.965e+04	6.094e+03	**6.533e+02**	7.857e+02	5.018e+02	+	**1.811e+04**	1.861e+04	5.227e+03	+	**2.708e+03**	2.837e+03	1.400e+03	+
f_5	30	5.243e+03	5.314e+03	1.033e+03	**2.470e+03**	2.490e+03	4.814e+02	+	**4.436e+02**	4.746e+02	3.038e+02	+	**2.569e+03**	2.614e+03	4.749e+02	+
	50	1.295e+04	1.262e+04	1.969e+03	**3.425e+03**	3.363e+03	5.542e+02	+	**4.249e+03**	4.278e+03	6.310e+02	+	**3.830e+03**	3.796e+03	4.907e+02	+
f_6	30	1.148e+01	2.796e+01	4.218e+01	**1.665e+01**	2.363e+01	2.879e+01	=	**9.000e−03**	5.609e−01	1.738e+00	+	**1.758e+01**	2.241e+01	2.147e+01	+
	50	4.016e+01	5.831e+01	4.576e+01	**3.806e+01**	5.808e+01	3.589e+01	=	**3.782e+01**	5.276e+01	2.660e+01	=	**3.897e+01**	5.049e+01	2.297e+01	=
f_7	30	4.696e+03	4.696e+03	5.039e−01	**4.696e+03**	4.696e+03	1.837e−12	=	**4.696e+03**	4.696e+03	1.837e−12	=	**4.696e+03**	4.696e+03	1.837e−12	=
	50	6.195e+03	6.197e+03	4.553e+00	**6.195e+03**	6.195e+03	4.594e−12	+	6.196e+03	6.196e+03	4.949e−01	−	**6.195e+03**	6.195e+03	4.594e−12	+
f_8	30	2.095e+01	2.094e+01	5.696e−02	**2.094e+01**	2.094e+01	5.119e−02	=	2.095e+01	2.095e+01	4.447e−02	=	2.096e+01	2.095e+01	5.207e−02	=
	50	2.115e+01	2.114e+01	2.978e−02	**2.114e+01**	2.114e+01	4.412e−02	=	**2.114e+01**	2.113e+01	3.845e−02	=	**2.114e+01**	2.113e+01	4.265e−02	=
f_9	30	5.622e+01	5.635e+01	1.079e+01	**3.980e+01**	4.107e+01	1.164e+01	+	**4.278e+01**	4.473e+01	1.369e+01	+	**4.179e+01**	4.171e+01	1.250e+01	+
	50	1.338e+02	1.329e+02	2.131e+01	**8.805e+01**	8.911e+01	1.861e+01	+	**9.999e+01**	9.657e+01	2.265e+01	+	**8.656e+01**	8.754e+01	2.055e+01	+
f_{10}	30	7.213e+01	7.556e+01	1.919e+01	**4.908e+01**	6.447e+01	4.302e+01	+	1.553e+02	1.419e+02	4.825e+01	−	1.033e+02	1.113e+02	6.642e+01	=
	50	**1.691e+02**	1.755e+02	4.101e+01	3.418e+02	2.837e+02	1.187e+02	−	3.100e+02	2.837e+02	1.024e+02	−	3.528e+02	3.404e+02	5.788e+01	−
f_{11}	30	2.790e+01	2.802e+01	1.926e+00	**2.872e+01**	2.737e+01	7.856e+00	=	3.503e+01	3.471e+01	3.749e+00	−	**2.575e+01**	2.604e+01	8.401e+00	=
	50	**5.483e+01**	5.471e+01	3.821e+00	6.843e+01	6.552e+01	9.517e+00	−	6.948e+01	6.915e+01	4.423e+00	−	6.922e+01	6.769e+01	5.956e+00	−
f_{12}	30	1.590e+03	2.585e+03	2.746e+03	**1.762e+03**	2.423e+03	2.512e+03	=	**1.247e+03**	2.217e+03	2.435e+03	=	1.970e+03	2.826e+03	3.003e+03	=
	50	1.447e+04	1.505e+04	9.270e+03	**8.694e+03**	1.056e+04	8.017e+03	+	1.665e+04	1.667e+04	1.244e+04	=	**4.577e+03**	7.914e+03	7.815e+03	+
f_{13}	30	6.252e+00	6.503e+00	1.855e+00	**3.049e+00**	3.164e+00	7.816e−01	+	**3.260e+00**	3.286e+00	8.615e−01	+	**3.294e+00**	3.376e+00	8.327e−01	+
	50	1.763e+01	1.839e+01	4.973e+00	**5.909e+00**	5.914e+00	1.052e+00	+	**6.739e+00**	7.813e+00	4.580e+00	+	**5.988e+00**	6.000e+00	1.184e+00	+
f_{14}	30	**1.237e+01**	1.229e+01	3.984e−01	1.300e+01	1.279e+01	5.299e−01	−	1.307e+01	1.299e+01	4.143e−01	−	1.295e+01	1.292e+01	3.989e−01	−
	50	**2.223e+01**	2.209e+01	4.659e−01	2.297e+01	2.288e+01	2.941e−01	−	2.301e+01	2.288e+01	3.969e−01	−	2.295e+01	2.291e+01	2.038e−01	−
f_{15}	30	**3.033e+02**	3.249e+02	8.537e+01	4.000e+02	3.663e+02	7.742e+01	−	4.000e+02	3.758e+02	8.810e+01	−	3.000e+02	3.477e+02	7.293e+01	=
	50	**2.511e+02**	2.834e+02	8.431e+01	4.000e+02	3.541e+02	8.385e+01	−	4.000e+02	3.563e+02	8.323e+01	−	4.000e+02	3.360e+02	9.207e+01	−
f_{16}	30	1.187e+02	1.356e+02	5.300e+01	**7.661e+01**	1.064e+02	8.450e+01	+	1.792e+02	1.719e+02	7.859e+01	−	1.004e+02	1.459e+02	1.164e+02	=
	50	**1.318e+02**	1.408e+02	3.960e+01	2.399e+02	2.025e+02	8.719e+01	−	2.095e+02	1.981e+02	6.608e+01	−	2.475e+02	2.417e+02	3.913e+01	−
f_{17}	30	1.580e+02	1.737e+02	6.599e+01	**8.545e+01**	1.348e+02	1.009e+02	+	1.851e+02	2.043e+02	8.026e+01	+	1.872e+02	1.837e+02	1.024e+02	=
	50	**2.038e+02**	2.099e+02	4.542e+01	2.602e+02	2.527e+02	5.254e+01	−	2.653e+02	2.620e+02	3.378e+01	−	2.620e+02	2.666e+02	4.111e+01	−
f_{18}	30	9.270e+02	9.101e+02	4.536e+01	**9.101e+02**	9.036e+02	2.658e+01	+	**9.071e+02**	8.988e+02	2.948e+01	+	**9.119e+02**	8.892e+02	4.808e+01	+
	50	9.928e+02	9.935e+02	2.367e+01	**9.279e+02**	9.255e+02	1.890e+01	+	**9.234e+02**	9.069e+02	9.336e+01	+	**9.280e+02**	9.182e+02	3.614e+01	+
f_{19}	30	9.247e+02	9.209e+02	3.222e+01	**9.103e+02**	8.950e+02	3.879e+01	+	**9.067e+02**	9.029e+02	2.133e+01	+	**9.120e+02**	9.016e+02	3.442e+01	+
	50	9.866e+02	9.913e+02	1.812e+01	**9.259e+02**	9.111e+02	9.017e+01	+	**9.229e+02**	9.031e+02	9.205e+01	+	**9.276e+02**	9.235e+02	2.762e+01	+
f_{20}	30	9.247e+02	9.167e+02	3.558e+01	**9.097e+02**	8.930e+02	4.106e+01	+	**9.076e+02**	9.054e+02	1.534e+01	+	**9.121e+02**	8.927e+02	4.399e+01	+
	50	9.868e+02	9.890e+02	2.113e+01	**9.270e+02**	9.193e+02	3.153e+01	+	**9.226e+02**	9.180e+02	2.445e+01	+	**9.267e+02**	9.131e+02	3.987e+01	+
f_{21}	30	5.000e+02	5.060e+02	4.243e+01	5.000e+02	5.248e+02	1.067e+02	=	5.000e+02	5.240e+02	8.221e+01	=	5.000e+02	5.512e+02	1.647e+02	=
	50	5.000e+02	5.199e+02	1.063e+02	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	5.000e+02	0.000e+00	=	5.000e+02	5.242e+02	8.277e+01	=
f_{22}	30	9.649e+02	9.675e+02	2.823e+01	**9.155e+02**	9.161e+02	1.481e+01	+	**9.201e+02**	9.157e+02	1.564e+01	+	**9.168e+02**	9.163e+02	1.373e+01	+
	50	1.006e+03	1.006e+03	1.564e+01	**9.648e+02**	9.623e+02	1.051e+01	+	**9.812e+02**	9.810e+02	1.259e+01	+	**9.626e+02**	9.612e+02	9.699e+00	+
f_{23}	30	5.000e+02	5.653e+02	1.887e+02	**5.000e+02**	5.180e+02	7.197e+01	=	**5.000e+02**	5.180e+02	7.197e+01	=	**5.000e+02**	5.321e+02	1.348e+02	=
	50	5.000e+02	5.000e+02	0.000e+00	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	5.120e+02	5.938e+01	=

(*continued on next page*)

Table 5 (*continued*)

f_i	D	BBPSO			BBPSO:DE/rand/1				BBPSO:DE/rand/2				BBPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	2.000e+02	2.000e+02	0.000e+00	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	2.000e+02	2.211e+02	1.374e+02	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
f_{25}	30	1.669e+03	1.668e+03	7.609e+00	**1.636e+03**	1.637e+03	7.052e+00	+	**1.658e+03**	1.657e+03	9.273e+00	+	**1.632e+03**	1.634e+03	6.186e+00	+
	50	1.724e+03	1.724e+03	6.689e+00	**1.686e+03**	1.687e+03	5.760e+00	+	**1.713e+03**	1.714e+03	4.454e+00	+	**1.683e+03**	1.683e+03	3.877e+00	+
					Total number of (+/−/=): (27/8/15)				Total number of (+/−/=): (23/13/14)				Total number of (+/−/=): (25/7/18)			

Table 6
Error values of the DMSPSO algorithm and their corresponding explorative hybrid DE variants on the 30 and 50–dimensional CEC 2005 benchmark functions.

f_i	D	DMSPSO			DMSPSO:DE/rand/1				DMSPSO:DE/rand/2				DMSPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	1.143e+02	3.135e+02	4.149e+02	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	2.349e+02	3.870e+02	3.855e+02	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
f_2	30	1.536e+02	7.801e+02	2.109e+03	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	4.000e−05	1.979e−04	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	3.311e+02	9.666e+02	1.409e+03	**7.500e−02**	1.199e−01	1.503e−01	+	2.365e+03	2.583e+03	1.134e+03	−	**1.426e+01**	1.863e+01	1.291e+01	+
f_3	30	3.898e+06	5.623e+06	6.225e+06	**4.907e+05**	5.242e+05	2.489e+05	+	**7.316e+05**	8.041e+05	4.326e+05	+	**6.509e+05**	7.501e+05	3.905e+05	+
	50	8.835e+06	1.317e+07	1.579e+07	**1.373e+06**	1.434e+06	4.832e+05	+	2.395e+07	2.499e+07	9.690e+06	−	**2.382e+06**	2.423e+06	9.628e+05	+
f_4	30	2.945e+02	8.561e+02	1.292e+03	**5.500e−03**	5.178e−02	1.071e−01	+	**1.770e−01**	6.673e+00	4.310e+01	+	**9.000e−03**	5.451e+00	3.003e+01	+
	50	1.306e+04	1.343e+04	3.945e+03	**3.866e+02**	5.254e+02	4.393e+02	+	**1.014e+04**	1.077e+04	3.905e+03	+	**1.586e+03**	1.653e+03	9.844e+02	+
f_5	30	4.368e+03	4.261e+03	1.873e+03	**2.283e+03**	2.296e+03	8.238e+02	+	**4.280e−01**	1.787e+02	6.176e+02	+	**2.550e+03**	2.590e+03	8.867e+02	+
	50	5.311e+03	5.533e+03	1.449e+03	**3.073e+03**	3.170e+03	7.913e+02	+	**3.207e+03**	3.150e+03	1.448e+03	+	**3.454e+03**	3.560e+03	6.465e+02	+
f_6	30	2.226e+06	2.721e+07	7.289e+07	**1.678e+01**	5.107e+01	8.251e+01	+	**8.445e−01**	1.503e+00	1.816e+00	+	**1.910e+01**	4.303e+01	5.599e+01	+
	50	2.226e+06	1.768e+07	4.103e+07	**5.984e+01**	1.031e+02	8.942e+01	+	**3.591e+01**	5.565e+01	3.536e+01	+	**4.329e+01**	7.337e+01	4.989e+01	+
f_7	30	4.297e+03	4.335e+03	2.190e+02	**4.186e+03**	4.208e+03	1.397e+02	+	**4.224e+03**	4.204e+03	1.629e+02	+	**4.214e+03**	4.220e+03	2.013e+02	+
	50	6.029e+03	6.050e+03	1.312e+02	**6.001e+03**	5.989e+03	5.232e+01	+	**5.973e+03**	5.963e+03	8.479e+01	+	**5.974e+03**	5.979e+03	5.398e+01	+
f_8	30	**2.093e+01**	2.093e+01	6.189e−02	2.097e+01	2.097e+01	4.510e−02	−	2.098e+01	2.098e+01	5.030e−02	−	2.098e+01	2.098e+01	4.104e−02	−
	50	**2.113e+01**	2.113e+01	3.770e−02	2.115e+01	2.114e+01	3.609e−02	=	2.116e+01	2.115e+01	3.617e−02	−	2.116e+01	2.115e+01	3.091e−02	−
f_9	30	4.524e+01	4.848e+01	1.499e+01	**3.096e+01**	3.194e+01	1.017e+01	+	**4.324e+01**	4.726e+01	1.254e+01	=	**2.639e+01**	2.798e+01	9.064e+00	+
	50	1.006e+02	9.893e+01	2.169e+01	**5.926e+01**	6.024e+01	1.518e+01	+	**9.372e+01**	9.706e+01	2.443e+01	=	**5.330e+01**	5.209e+01	1.215e+01	+
f_{10}	30	7.797e+01	7.999e+01	2.003e+01	**5.940e+01**	5.972e+01	2.586e+01	+	9.556e+01	9.700e+01	1.619e+01	−	8.544e+01	8.464e+01	1.613e+01	=
	50	**1.673e+02**	1.662e+02	2.167e+01	1.977e+02	1.966e+02	1.990e+01	−	2.255e+02	2.209e+02	2.477e+01	−	1.899e+02	1.874e+02	1.633e+01	−
f_{11}	30	2.929e+01	2.903e+01	2.259e+00	**2.981e+01**	2.787e+01	5.862e+00	=	3.315e+01	3.297e+01	1.829e+00	−	3.150e+01	3.089e+01	2.397e+00	−
	50	**5.795e+01**	5.774e+01	2.256e+00	6.084e+01	6.083e+01	1.433e+00	−	6.385e+01	6.346e+01	2.902e+00	−	6.102e+01	6.085e+01	2.349e+00	−
f_{12}	30	5.375e+04	7.843e+04	6.836e+04	**2.957e+04**	4.452e+04	4.304e+04	+	**2.522e+04**	5.098e+04	7.114e+04	+	**1.866e+04**	3.142e+04	3.786e+04	+
	50	1.212e+05	1.659e+05	1.461e+05	**6.368e+04**	7.716e+04	7.845e+04	+	**6.368e+04**	8.903e+04	7.232e+04	+	**5.644e+04**	7.678e+04	7.333e+04	+
f_{13}	30	9.962e+00	1.127e+01	5.622e+00	**2.981e+00**	3.039e+00	8.358e−01	+	**6.792e+00**	6.786e+00	1.972e+00	+	**3.251e+00**	3.827e+00	1.716e+00	+
	50	7.812e+00	8.219e+00	2.218e+00	**5.843e+00**	6.220e+00	1.726e+00	+	1.487e+01	1.490e+01	4.858e+00	−	1.128e+01	1.163e+01	5.020e+00	−
f_{14}	30	**1.212e+01**	1.207e+01	6.594e−01	1.294e+01	1.289e+01	2.232e−01	−	1.314e+01	1.306e+01	2.150e−01	−	1.294e+01	1.285e+01	3.322e−01	−
	50	**2.246e+01**	2.241e+01	2.289e−01	2.278e+01	2.276e+01	1.869e−01	−	2.272e+01	2.274e+01	2.212e−01	−	2.277e+01	2.272e+01	2.164e−01	−
f_{15}	30	5.105e+02	5.222e+02	8.016e+01	**3.277e+02**	3.364e+02	9.731e+01	+	**2.767e+02**	2.926e+02	8.687e+01	+	**3.047e+02**	3.248e+02	7.782e+01	+
	50	3.694e+02	3.778e+02	6.679e+01	**2.265e+02**	2.479e+02	6.049e+01	+	**2.784e+02**	2.913e+02	5.926e+01	+	**2.143e+02**	2.401e+02	6.021e+01	+
f_{16}	30	2.250e+02	2.916e+02	1.683e+02	**1.109e+02**	1.553e+02	1.157e+02	+	**1.332e+02**	1.753e+02	9.790e+01	+	**1.114e+02**	1.650e+02	1.174e+02	+
	50	1.419e+02	1.670e+02	8.193e+01	**1.495e+02**	1.614e+02	3.348e+01	+	1.730e+02	1.869e+02	4.981e+01	−	1.502e+02	1.789e+02	7.482e+01	−
f_{17}	30	2.300e+02	2.988e+02	1.675e+02	**1.653e+02**	2.035e+02	1.136e+02	+	**1.865e+02**	2.311e+02	9.230e+01	=	**1.817e+02**	2.310e+02	1.113e+02	=
	50	**2.163e+02**	2.413e+02	7.257e+01	2.281e+02	2.471e+02	5.777e+01	−	2.472e+02	2.579e+02	4.811e+01	−	**2.154e+02**	2.411e+02	6.038e+01	=
f_{18}	30	9.249e+02	9.372e+02	3.011e+01	**9.059e+02**	9.070e+02	2.901e+00	+	**9.069e+02**	9.072e+02	2.548e+00	+	**9.068e+02**	9.076e+02	4.379e+00	+
	50	9.300e+02	9.334e+02	1.156e+01	**9.170e+02**	9.200e+02	7.908e+00	+	**9.146e+02**	9.167e+02	6.179e+00	+	**9.185e+02**	9.200e+02	5.550e+00	+
f_{19}	30	9.191e+02	9.328e+02	2.703e+01	**9.068e+02**	9.069e+02	2.713e+00	+	**9.060e+02**	9.064e+02	1.991e+00	+	**9.074e+02**	9.083e+02	5.963e+00	+
	50	9.297e+02	9.315e+02	9.374e+00	**9.158e+02**	9.169e+02	4.131e+00	+	**9.153e+02**	9.166e+02	4.368e+00	+	**9.202e+02**	9.205e+02	4.918e+00	+
f_{20}	30	9.281e+02	9.394e+02	2.991e+01	**9.068e+02**	9.087e+02	8.649e+00	+	**9.053e+02**	9.073e+02	5.078e+00	+	**9.069e+02**	9.070e+02	2.187e+00	+
	50	9.299e+02	9.303e+02	8.262e+00	**9.170e+02**	9.184e+02	5.230e+00	+	**9.149e+02**	9.165e+02	5.067e+00	+	**9.194e+02**	9.194e+02	5.415e+00	+
f_{21}	30	1.098e+03	1.037e+03	1.558e+02	**1.091e+03**	9.962e+02	2.188e+02	+	1.091e+03	1.090e+03	5.062e+00	−	**1.088e+03**	9.138e+02	2.702e+02	+
	50	1.011e+03	1.011e+03	2.700e+00	**1.008e+03**	1.009e+03	3.169e+00	+	**1.008e+03**	1.009e+03	3.548e+00	+	**1.009e+03**	1.009e+03	3.347e+00	+
f_{22}	30	9.261e+02	9.344e+02	5.295e+01	**8.737e+02**	8.730e+02	2.520e+01	+	**8.569e+02**	8.644e+02	2.003e+01	+	**8.695e+02**	8.702e+02	2.505e+01	+
	50	8.997e+02	9.059e+02	1.972e+01	**8.960e+02**	8.983e+02	8.919e+00	=	**8.997e+02**	9.011e+02	8.508e+00	=	**8.960e+02**	8.980e+02	9.710e+00	+
f_{23}	30	1.097e+03	1.063e+03	1.192e+02	**1.089e+03**	1.013e+03	1.957e+02	+	1.091e+03	1.089e+03	6.121e+00	−	**1.082e+03**	8.302e+02	2.894e+02	+
	50	1.011e+03	1.011e+03	2.830e+00	**1.008e+03**	1.009e+03	4.458e+00	+	**1.010e+03**	1.010e+03	3.895e+00	=	**1.009e+03**	1.009e+03	2.799e+00	+

(*continued on next page*)

Table 6 (*continued*)

f_i	D	DMSPSO			DMSPSO:DE/rand/1				DMSPSO:DE/rand/2				DMSPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	9.769e+02	9.851e+02	5.687e+01	**9.624e+02**	9.588e+02	2.023e+01	+	**9.667e+02**	9.605e+02	1.655e+01	+	**9.609e+02**	9.571e+02	2.019e+01	+
	50	1.025e+03	1.025e+03	2.067e+00	**1.026e+03**	1.009e+03	1.168e+02	+	**1.023e+03**	1.022e+03	6.450e+00	+	**1.026e+03**	9.928e+02	1.635e+02	=
f_{25}	30	1.639e+03	1.640e+03	8.368e+00	**1.634e+03**	1.634e+03	5.722e+00	+	**1.636e+03**	1.635e+03	4.445e+00	+	**1.635e+03**	1.634e+03	6.052e+00	+
	50	1.675e+03	1.676e+03	4.880e+00	**1.674e+03**	1.673e+03	5.170e+00	=	1.676e+03	1.677e+03	3.927e+00	–	**1.674e+03**	1.673e+03	4.884e+00	=
					Total number of (+/–/=): (40/6/4)				Total number of (+/–/=): (29/16/5)				Total number of (+/–/=): (36/9/5)			

Table 7
Error values of the FIPS algorithm and their corresponding explorative hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions.

f_i	D	FIPS			FIPS:DE/rand/1				FIPS:DE/rand/2				FIPS:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	3.185e+02	5.252e+02	5.571e+02	**0.000e+00**	3.832e−02	2.647e−01	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	1.149e+03	1.673e+03	1.524e+03	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
f_2	30	1.460e+04	1.470e+04	2.316e+03	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	2.633e+04	2.574e+04	4.424e+03	**0.000e+00**	1.580e−03	5.218e−03	+	**0.000e+00**	1.140e−03	7.492e−03	+	**0.000e+00**	6.860e−03	4.111e−02	+
f_3	30	1.530e+07	1.945e+07	1.109e+07	**3.881e+05**	4.428e+05	2.337e+05	+	**5.373e+05**	6.079e+05	2.831e+05	+	**3.810e+05**	4.056e+05	1.826e+05	+
	50	5.586e+07	5.867e+07	2.346e+07	**6.488e+05**	9.426e+05	1.479e+06	+	**4.667e+06**	5.565e+06	4.507e+06	+	**9.555e+05**	1.079e+06	5.859e+05	+
f_4	30	2.077e+04	2.068e+04	3.107e+03	**4.952e+03**	4.871e+03	2.180e+03	+	**3.581e+01**	2.442e+02	6.464e+02	+	**7.306e+03**	7.655e+03	2.637e+03	+
	50	3.355e+04	3.424e+04	3.850e+03	**1.220e+04**	1.165e+04	4.941e+03	+	**1.306e+03**	1.641e+03	1.231e+03	+	**2.004e+04**	1.931e+04	4.259e+03	+
f_5	30	1.164e+04	1.174e+04	1.393e+03	**3.398e+03**	3.429e+03	7.359e+02	+	**5.186e+03**	5.143e+03	8.013e+02	+	**3.619e+03**	3.538e+03	5.916e+02	+
	50	1.594e+04	1.589e+04	1.147e+03	**4.519e+03**	4.543e+03	9.469e+02	+	**7.063e+03**	6.949e+03	9.818e+02	+	**4.657e+03**	4.727e+03	6.627e+02	+
f_6	30	9.832e+06	2.457e+07	3.493e+07	**7.246e+01**	2.963e+02	6.637e+02	+	**1.361e+01**	1.268e+01	1.095e+01	+	**1.939e+01**	3.597e+01	4.064e+01	+
	50	6.483e+07	8.021e+07	6.118e+07	**1.067e+02**	1.364e+02	1.299e+02	+	**4.074e+01**	5.653e+01	2.737e+01	+	**6.222e+01**	8.810e+01	6.819e+01	+
f_7	30	7.507e+03	7.477e+03	2.158e+02	**6.989e+03**	6.928e+03	2.364e+02	+	**6.729e+03**	6.704e+03	1.477e+02	+	**6.865e+03**	6.842e+03	2.133e+02	+
	50	1.037e+04	1.036e+04	2.122e+02	**9.271e+03**	9.266e+03	2.407e+02	+	**9.071e+03**	9.055e+03	1.651e+02	+	**9.228e+03**	9.253e+03	2.641e+02	+
f_8	30	**2.095e+01**	2.094e+01	6.409e−02	2.098e+01	2.097e+01	5.124e−02	−	2.100e+01	2.099e+01	5.169e−02	−	2.099e+01	2.098e+01	6.824e−02	−
	50	**2.115e+01**	2.114e+01	4.304e−02	2.116e+01	2.115e+01	3.071e−02	=	2.115e+01	2.115e+01	4.101e−02	=	2.116e+01	2.115e+01	3.189e−02	=
f_9	30	5.401e+01	5.395e+01	1.097e+01	**2.935e+01**	2.973e+01	7.191e+00	+	**2.637e+01**	2.854e+01	6.386e+00	+	**2.686e+01**	2.698e+01	6.635e+00	+
	50	1.550e+02	1.530e+02	1.791e+01	**6.368e+01**	6.632e+01	9.660e+00	+	**7.263e+01**	7.315e+01	1.040e+01	+	**5.870e+01**	6.133e+01	1.013e+01	+
f_{10}	30	1.545e+02	1.525e+02	2.533e+01	**3.333e+01**	3.277e+01	8.352e+00	+	**3.234e+01**	3.544e+01	1.695e+01	+	**3.134e+01**	3.086e+01	6.779e+00	+
	50	3.868e+02	3.928e+02	3.625e+01	**7.164e+01**	7.174e+01	1.459e+01	+	**8.159e+01**	8.501e+01	1.440e+01	+	**7.014e+01**	7.132e+01	1.417e+01	+
f_{11}	30	2.662e+01	2.688e+01	2.641e+00	**1.137e+01**	1.119e+01	2.168e+00	+	**1.159e+01**	1.545e+01	9.702e+00	+	**1.185e+01**	1.196e+01	2.325e+00	+
	50	5.363e+01	5.343e+01	3.789e+00	**2.274e+01**	2.440e+01	7.502e+00	+	**5.923e+01**	4.992e+01	1.998e+01	+	**2.387e+01**	2.437e+01	3.438e+00	+
f_{12}	30	4.679e+04	5.185e+04	3.213e+04	**3.239e+03**	4.036e+03	3.810e+03	+	**1.299e+03**	2.104e+03	2.493e+03	+	**3.066e+03**	3.670e+03	3.464e+03	+
	50	2.771e+05	2.929e+05	1.490e+05	**1.694e+04**	1.672e+04	1.033e+04	+	**1.392e+04**	1.364e+04	1.144e+04	+	**1.263e+04**	1.408e+04	1.134e+04	+
f_{13}	30	9.604e+00	9.641e+00	1.730e+00	**2.788e+00**	2.903e+00	6.031e−01	+	**2.910e+00**	3.192e+00	1.411e+00	+	**2.797e+00**	2.949e+00	6.968e−01	+
	50	2.583e+01	2.638e+01	3.451e+00	**4.883e+00**	4.848e+00	8.670e−01	+	**4.659e+00**	5.496e+00	2.803e+00	+	**4.594e+00**	4.694e+00	8.625e−01	+
f_{14}	30	1.237e+01	1.234e+01	3.198e−01	**1.148e+01**	1.143e+01	4.640e−01	+	**1.168e+01**	1.167e+01	4.321e−01	+	**1.151e+01**	1.147e+01	3.986e−01	+
	50	2.196e+01	2.190e+01	3.078e−01	**2.086e+01**	2.087e+01	4.916e−01	+	**2.155e+01**	2.147e+01	4.193e−01	+	**2.093e+01**	2.078e+01	5.681e−01	+
f_{15}	30	4.536e+02	4.556e+02	8.414e+01	**3.149e+02**	3.472e+02	7.027e+01	+	**4.000e+02**	3.754e+02	6.530e+01	+	**3.119e+02**	3.445e+02	5.427e+01	+
	50	5.175e+02	5.174e+02	5.406e+01	**4.000e+02**	3.430e+02	8.769e+01	+	**4.000e+02**	3.630e+02	7.620e+01	+	**4.000e+02**	3.551e+02	7.709e+01	+
f_{16}	30	3.271e+02	3.414e+02	1.081e+02	**6.025e+01**	1.458e+02	1.688e+02	+	**6.798e+01**	1.380e+02	1.339e+02	+	**6.027e+01**	1.740e+02	1.904e+02	+
	50	3.227e+02	3.296e+02	6.967e+01	**5.036e+01**	7.310e+01	8.493e+01	+	**5.663e+01**	6.513e+01	3.390e+01	+	**4.877e+01**	7.650e+01	9.686e+01	+
f_{17}	30	4.333e+02	4.502e+02	1.420e+02	**7.310e+01**	1.562e+02	1.659e+02	+	**6.372e+01**	1.164e+02	1.052e+02	+	**6.647e+01**	1.700e+02	1.842e+02	+
	50	4.371e+02	4.576e+02	9.022e+01	**6.467e+01**	9.462e+01	1.009e+02	+	**6.479e+01**	7.626e+01	5.689e+01	+	**6.394e+01**	9.427e+01	1.014e+02	+
f_{18}	30	1.055e+03	1.052e+03	2.219e+01	**9.662e+02**	9.315e+02	7.552e+01	+	**9.706e+02**	9.313e+02	7.494e+01	+	**9.630e+02**	9.288e+02	7.394e+01	+
	50	1.070e+03	1.069e+03	1.346e+01	**9.982e+02**	9.848e+02	3.978e+01	+	**1.002e+03**	9.840e+02	6.248e+01	+	**9.787e+02**	9.660e+02	4.164e+01	+
f_{19}	30	1.047e+03	1.049e+03	1.873e+01	**9.694e+02**	9.385e+02	7.080e+01	+	**9.729e+02**	9.517e+02	6.263e+01	+	**9.621e+02**	9.393e+02	6.294e+01	+
	50	1.070e+03	1.070e+03	1.603e+01	**9.849e+02**	9.720e+02	4.461e+01	+	**1.004e+03**	9.856e+02	6.329e+01	+	**9.827e+02**	9.540e+02	9.002e+01	+
f_{20}	30	1.049e+03	1.050e+03	1.811e+01	**9.674e+02**	9.316e+02	7.572e+01	+	**9.691e+02**	9.357e+02	7.328e+01	+	**9.583e+02**	9.262e+02	7.171e+01	+
	50	1.062e+03	1.065e+03	1.335e+01	**9.944e+02**	9.836e+02	3.583e+01	+	**9.990e+02**	9.781e+02	6.715e+01	+	**9.865e+02**	9.744e+02	3.637e+01	+
f_{21}	30	1.172e+03	1.093e+03	1.633e+02	**5.000e+02**	5.130e+02	9.193e+01	+	**5.000e+02**	5.187e+02	9.839e+01	+	**5.000e+02**	5.194e+02	1.007e+02	+
	50	1.209e+03	1.196e+03	5.787e+01	**5.000e+02**	5.645e+02	1.761e+02	+	**5.000e+02**	5.000e+02	0.000e+00	+	**5.000e+02**	5.720e+02	1.936e+02	+
f_{22}	30	1.121e+03	1.126e+03	2.782e+01	**9.995e+02**	9.999e+02	1.781e+01	+	**1.006e+03**	1.004e+03	1.734e+01	+	**9.987e+02**	1.000e+03	1.827e+01	+
	50	1.200e+03	1.202e+03	2.191e+01	**1.081e+03**	1.082e+03	1.643e+01	+	**1.074e+03**	1.077e+03	1.170e+01	+	**1.077e+03**	1.077e+03	1.523e+01	+
f_{23}	30	1.177e+03	1.072e+03	1.936e+02	**5.000e+02**	5.726e+02	1.914e+02	+	**5.000e+02**	5.381e+02	1.398e+02	+	**5.000e+02**	5.205e+02	9.989e+01	+
	50	1.210e+03	1.191e+03	9.395e+01	**5.000e+02**	5.809e+02	2.213e+02	+	**5.000e+02**	5.545e+02	1.868e+02	+	**5.000e+02**	5.862e+02	2.213e+02	+

(*continued on next page*)

Table 7 (*continued*)

f_i	D	FIPS			FIPS:DE/rand/1				FIPS:DE/rand/2				FIPS:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	1.268e+03	1.254e+03	5.409e+01	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+
	50	1.282e+03	1.283e+03	1.049e+01	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+
f_{25}	30	1.780e+03	1.781e+03	1.046e+01	**1.749e+03**	1.747e+03	1.070e+01	+	**1.749e+03**	1.747e+03	7.632e+00	+	**1.744e+03**	1.744e+03	1.096e+01	+
	50	1.866e+03	1.866e+03	7.076e+00	**1.815e+03**	1.814e+03	8.022e+00	+	**1.807e+03**	1.807e+03	6.792e+00	+	**1.812e+03**	1.812e+03	8.940e+00	+
					Total number of (+/−/=): (48/1/1)				Total number of (+/−/=): (48/1/1)				Total number of (+/−/=): (48/1/1)			

Table 8
Error values of the UPSO algorithm and their corresponding explorative hybrid DE variants on the 30 and 50–dimensional CEC 2005 benchmark functions.

f_i	D	UPSO			UPSO:DE/rand/1				UPSO:DE/rand/2				UPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	1.269e+03	1.306e+03	7.328e+02	**0.000e+00**	1.115e+01	3.569e+01	+	**0.000e+00**	5.600e−04	3.818e−03	+	**0.000e+00**	4.000e−05	2.828e−04	+
	50	6.840e+02	7.100e+02	3.290e+02	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	4.000e−05	1.979e−04	+	**0.000e+00**	0.000e+00	0.000e+00	+
f_2	30	6.688e+03	7.602e+03	5.290e+03	**0.000e+00**	5.032e−02	3.380e−01	+	**0.000e+00**	1.020e−03	2.190e−03	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	3.632e+03	4.220e+03	2.894e+03	**0.000e+00**	0.000e+00	0.000e+00	+	**3.450e−01**	6.988e−01	1.017e+00	+	**0.000e+00**	0.000e+00	0.000e+00	+
f_3	30	4.308e+07	5.303e+07	3.856e+07	**4.715e+05**	5.827e+05	3.593e+05	+	**7.485e+05**	1.103e+06	1.135e+06	+	**5.068e+05**	5.212e+05	2.279e+05	+
	50	4.885e+07	5.340e+07	3.743e+07	**1.922e+05**	2.047e+05	1.143e+05	+	**1.098e+06**	1.442e+06	1.337e+06	+	**4.470e+05**	5.147e+05	2.627e+05	+
f_4	30	1.913e+04	1.876e+04	6.086e+03	**1.246e+01**	6.605e+01	1.234e+02	+	**1.450e+00**	2.466e+00	2.450e+00	+	**9.230e−01**	2.132e+00	2.568e+00	+
	50	1.353e+04	1.449e+04	4.462e+03	**0.000e+00**	1.340e−03	5.009e−03	+	**1.261e+01**	2.093e+01	3.546e+01	+	**3.400e−02**	7.682e−02	1.300e−01	+
f_5	30	1.268e+04	1.282e+04	2.288e+03	**4.335e+03**	4.602e+03	1.181e+03	+	**8.583e+03**	8.398e+03	1.048e+03	+	**4.486e+03**	4.799e+03	1.273e+03	+
	50	1.177e+04	1.207e+04	2.304e+03	**6.300e+03**	6.330e+03	1.679e+03	+	**7.946e+03**	7.927e+03	1.067e+03	+	**5.411e+03**	5.738e+03	1.690e+03	+
f_6	30	6.826e+06	1.187e+07	1.355e+07	**8.930e+01**	8.370e+04	4.973e+05	+	**5.959e+00**	5.795e+00	3.843e+00	+	**2.160e+01**	3.938e+01	4.717e+01	+
	50	1.160e+06	2.731e+06	3.667e+06	**1.848e+01**	2.034e+01	2.199e+01	+	**9.178e+00**	1.485e+01	1.959e+01	+	**8.322e+00**	8.388e+00	9.917e+00	+
f_7	30	7.513e+03	7.524e+03	3.409e+02	7.813e+03	7.802e+03	2.404e+02	−	**7.325e+03**	7.286e+03	2.317e+02	+	7.829e+03	7.812e+03	2.308e+02	−
	50	7.419e+03	7.420e+03	3.034e+02	7.725e+03	7.713e+03	1.853e+02	−	**7.134e+03**	7.091e+03	1.528e+02	+	7.774e+03	7.739e+03	2.122e+02	−
f_8	30	**2.096e+01**	2.095e+01	5.009e−02	2.098e+01	2.098e+01	4.811e−02	−	2.100e+01	2.098e+01	5.656e−02	−	2.097e+01	2.096e+01	5.457e−02	=
	50	**2.094e+01**	2.093e+01	5.023e−02	2.097e+01	2.096e+01	4.350e−02	−	2.096e+01	2.095e+01	5.223e−02	−	2.096e+01	2.094e+01	5.685e−02	=
f_9	30	7.719e+01	7.839e+01	1.689e+01	**4.079e+01**	4.047e+01	1.103e+01	+	**5.671e+01**	5.819e+01	1.534e+01	+	**3.930e+01**	4.167e+01	1.408e+01	+
	50	6.326e+01	6.520e+01	1.764e+01	**4.079e+01**	4.209e+01	1.191e+01	+	**5.572e+01**	5.717e+01	1.477e+01	+	**3.880e+01**	3.761e+01	1.051e+01	+
f_{10}	30	1.526e+02	1.589e+02	5.514e+01	**4.129e+01**	4.673e+01	2.823e+01	+	**1.295e+02**	1.118e+02	6.239e+01	+	**4.676e+01**	9.200e+01	7.336e+01	+
	50	1.453e+02	1.442e+02	4.564e+01	**3.731e+01**	7.085e+01	6.409e+01	+	1.532e+02	1.284e+02	6.151e+01	=	**5.273e+01**	9.895e+01	6.778e+01	+
f_{11}	30	3.164e+01	3.140e+01	4.692e+00	**1.455e+01**	1.666e+01	7.222e+00	+	3.579e+01	2.791e+01	1.310e+01	=	**1.417e+01**	1.557e+01	5.399e+00	+
	50	2.910e+01	2.954e+01	4.439e+00	**9.605e+00**	1.125e+01	8.024e+00	+	3.621e+01	3.014e+01	1.232e+01	−	**7.917e+00**	1.132e+01	9.895e+00	+
f_{12}	30	7.752e+04	8.984e+04	5.430e+04	**2.678e+03**	4.814e+03	5.487e+03	+	**1.989e+03**	2.706e+03	3.458e+03	+	**1.992e+03**	3.049e+03	3.232e+03	+
	50	6.052e+04	7.135e+04	4.785e+04	**1.891e+03**	3.180e+03	4.053e+03	+	**1.500e+03**	3.142e+03	4.370e+03	+	**1.429e+03**	2.281e+03	3.023e+03	+
f_{13}	30	8.489e+00	9.231e+00	4.563e+00	**3.091e+00**	3.330e+00	8.610e−01	+	**3.258e+00**	3.456e+00	1.053e+00	+	**3.194e+00**	3.304e+00	1.034e+00	+
	50	6.236e+00	6.478e+00	2.102e+00	**2.846e+00**	2.946e+00	7.385e−01	+	**3.154e+00**	3.365e+00	1.071e+00	+	**3.019e+00**	3.215e+00	8.605e−01	+
f_{14}	30	1.289e+01	1.283e+01	4.178e−01	**1.268e+01**	1.253e+01	6.116e−01	+	1.300e+01	1.292e+01	4.140e−01	=	**1.271e+01**	1.254e+01	6.114e−01	+
	50	1.282e+01	1.274e+01	4.817e−01	**1.277e+01**	1.271e+01	4.678e−01	=	1.297e+01	1.289e+01	4.299e−01	=	**1.266e+01**	1.254e+01	4.786e−01	+
f_{15}	30	5.373e+02	5.305e+02	9.060e+01	**3.255e+02**	3.527e+02	9.148e+01	+	**4.000e+02**	3.523e+02	8.342e+01	+	**3.110e+02**	3.420e+02	9.342e+01	+
	50	5.087e+02	4.843e+02	1.137e+02	**4.000e+02**	3.777e+02	9.675e+01	+	**4.000e+02**	3.901e+02	8.137e+01	+	**3.000e+02**	3.402e+02	8.983e+01	+
f_{16}	30	3.827e+02	3.865e+02	1.416e+02	**8.047e+01**	1.546e+02	1.503e+02	+	**2.493e+02**	2.713e+02	8.026e+01	+	**1.476e+02**	1.918e+02	1.473e+02	+
	50	2.909e+02	3.287e+02	1.336e+02	**2.138e+02**	2.171e+02	1.296e+02	+	**2.368e+02**	2.529e+02	1.042e+02	+	**2.337e+02**	2.777e+02	1.099e+02	+
f_{17}	30	3.486e+02	3.875e+02	1.322e+02	**1.241e+02**	1.678e+02	1.301e+02	+	**2.669e+02**	3.006e+02	8.620e+01	+	**2.520e+02**	2.903e+02	1.380e+02	+
	50	3.100e+02	3.497e+02	1.259e+02	**2.456e+02**	2.710e+02	1.000e+02	+	**2.677e+02**	2.920e+02	1.032e+02	+	**2.438e+02**	3.080e+02	1.035e+02	=
f_{18}	30	1.051e+03	1.047e+03	3.627e+01	**1.003e+03**	9.833e+02	7.023e+01	+	**1.009e+03**	1.002e+03	4.283e+01	+	**9.924e+02**	9.627e+02	7.874e+01	+
	50	1.035e+03	1.033e+03	3.041e+01	**1.022e+03**	1.006e+03	5.580e+01	+	**9.966e+02**	9.928e+02	2.924e+01	+	**1.018e+03**	9.866e+02	8.327e+01	+
f_{19}	30	1.040e+03	1.049e+03	4.636e+01	**1.002e+03**	9.740e+02	7.272e+01	+	**1.008e+03**	9.988e+02	5.215e+01	+	**9.933e+02**	9.567e+02	8.142e+01	+
	50	1.027e+03	1.028e+03	3.344e+01	**1.016e+03**	9.852e+02	8.262e+01	+	**9.948e+02**	9.961e+02	9.678e+00	+	**1.015e+03**	9.896e+02	7.872e+01	+
f_{20}	30	1.043e+03	1.044e+03	2.626e+01	**9.900e+02**	9.617e+02	7.339e+01	+	**1.012e+03**	9.822e+02	7.535e+01	+	**9.940e+02**	9.721e+02	6.738e+01	+
	50	1.027e+03	1.026e+03	2.788e+01	**1.017e+03**	9.980e+02	6.876e+01	+	**9.960e+02**	9.969e+02	1.033e+01	+	**1.021e+03**	9.903e+02	7.944e+01	=
f_{21}	30	1.162e+03	1.021e+03	2.099e+02	**5.000e+02**	5.127e+02	5.931e+01	+	**5.000e+02**	5.120e+02	5.938e+01	+	**5.001e+02**	5.663e+02	1.709e+02	+
	50	7.525e+02	8.542e+02	2.499e+02	**5.000e+02**	5.000e+02	5.051e−04	+	**5.000e+02**	5.240e+02	8.221e+01	+	**5.000e+02**	5.248e+02	1.066e+02	+
f_{22}	30	1.106e+03	1.109e+03	4.488e+01	**1.057e+03**	1.054e+03	3.798e+01	+	**1.038e+03**	1.040e+03	2.096e+01	+	**1.039e+03**	1.035e+03	2.602e+01	+
	50	1.085e+03	1.086e+03	4.747e+01	**1.055e+03**	1.053e+03	3.036e+01	+	**1.028e+03**	1.031e+03	1.829e+01	+	**1.046e+03**	1.052e+03	2.912e+01	+
f_{23}	30	1.156e+03	1.030e+03	2.002e+02	**5.000e+02**	5.411e+02	1.448e+02	+	**5.000e+02**	5.000e+02	2.399e−04	+	**5.002e+02**	5.799e+02	1.881e+02	+
	50	1.049e+03	9.335e+02	2.551e+02	**5.000e+02**	5.060e+02	4.243e+01	+	**5.000e+02**	5.060e+02	4.243e+01	+	**5.000e+02**	5.060e+02	4.243e+01	+

(*continued on next page*)

Table 8 (*continued*)

f_i	D	UPSO			UPSO:DE/rand/1				UPSO:DE/rand/2				UPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	1.141e+03	9.806e+02	3.183e+02	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+
	50	5.298e+02	6.952e+02	3.888e+02	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+	**2.000e+02**	2.000e+02	0.000e+00	+
f_{25}	30	1.778e+03	1.778e+03	1.256e+01	**1.769e+03**	1.767e+03	1.031e+01	+	**1.762e+03**	1.761e+03	9.534e+00	+	**1.765e+03**	1.763e+03	1.048e+01	+
	50	1.769e+03	1.771e+03	1.389e+01	1.770e+03	1.769e+03	9.494e+00	=	**1.759e+03**	1.758e+03	6.791e+00	+	1.773e+03	1.771e+03	9.548e+00	=
					Total number of (+/−/=): (44/4/2)				Total number of (+/−/=): (43/3/4)				Total number of (+/−/=): (43/2/5)			

Table 9
Error values of the CLPSO algorithm and their corresponding explorative hybrid DE variants on the 30 and 50–dimensional CEC 2005 benchmark functions.

f_i	D	CLPSO			CLPSO:DE/rand/1				CLPSO:DE/rand/2				CLPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_1	30	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=
	50	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=	0.000e+00	0.000e+00	0.000e+00	=
f_2	30	3.828e+02	3.828e+02	1.060e+02	**0.000e+00**	0.000e+00	0.000e+00	+	**0.000e+00**	1.000e−04	3.030e−04	+	**0.000e+00**	0.000e+00	0.000e+00	+
	50	1.013e+04	1.021e+04	1.357e+03	**2.100e−02**	3.002e−02	3.293e−02	+	1.014e+04	9.933e+03	2.373e+03	=	**1.027e+01**	1.128e+01	7.425e+00	+
f_3	30	1.204e+07	1.188e+07	3.107e+06	**3.336e+05**	3.406e+05	1.339e+05	+	**4.047e+05**	4.646e+05	2.456e+05	+	**3.712e+05**	4.149e+05	1.674e+05	+
	50	5.084e+07	4.930e+07	1.161e+07	**6.927e+05**	6.661e+05	2.143e+05	+	5.399e+07	5.356e+07	1.281e+07	=	**1.249e+06**	1.439e+06	6.393e+05	+
f_4	30	5.481e+03	5.396e+03	1.250e+03	**7.700e−02**	3.318e−01	8.563e−01	+	**9.700e−01**	1.456e+00	1.583e+00	+	**4.900e−02**	1.709e−01	2.517e−01	+
	50	3.477e+04	3.428e+04	5.637e+03	**2.968e+02**	3.415e+02	2.116e+02	+	**3.341e+04**	3.254e+04	5.068e+03	=	**1.563e+03**	1.704e+03	6.936e+02	+
f_5	30	4.011e+03	4.001e+03	4.276e+02	**2.712e+03**	2.683e+03	5.203e+02	+	**1.213e+03**	1.277e+03	5.506e+02	+	**2.937e+03**	3.060e+03	6.308e+02	+
	50	9.753e+03	9.698e+03	7.903e+02	**3.064e+03**	3.070e+03	4.047e+02	+	**8.794e+03**	8.800e+03	7.676e+02	+	**3.888e+03**	3.895e+03	5.917e+02	+
f_6	30	7.369e+00	1.779e+01	2.285e+01	1.102e+01	3.727e+01	6.968e+01	=	**1.340e−01**	5.998e−01	1.191e+00	+	7.037e+01	6.755e+01	5.690e+01	−
	50	8.998e+01	8.705e+01	3.757e+01	**4.090e+01**	6.993e+01	6.255e+01	+	**4.240e+01**	5.339e+01	2.736e+01	+	**4.263e+01**	6.680e+01	3.517e+01	+
f_7	30	**4.696e+03**	4.696e+03	1.837e−12	4.696e+03	4.696e+03	1.837e−12	=	4.696e+03	4.696e+03	1.837e−12	=	4.696e+03	4.696e+03	1.837e−12	=
	50	**6.195e+03**	6.195e+03	4.594e−12	6.195e+03	6.195e+03	4.594e−12	=	6.195e+03	6.195e+03	5.345e−03	−	6.195e+03	6.195e+03	4.594e−12	=
f_8	30	**2.072e+01**	2.072e+01	5.905e−02	2.098e+01	2.097e+01	5.312e−02	−	2.097e+01	2.096e+01	7.255e−02	−	2.098e+01	2.098e+01	5.331e−02	−
	50	**2.105e+01**	2.104e+01	4.617e−02	2.117e+01	2.116e+01	4.574e−02	−	2.117e+01	2.116e+01	3.962e−02	−	2.118e+01	2.117e+01	3.914e−02	−
f_9	30	**0.000e+00**	0.000e+00	0.000e+00	1.990e+01	2.020e+01	1.056e+01	−	2.436e+00	2.792e+00	1.584e+00	−	2.479e+00	5.407e+00	7.859e+00	−
	50	**0.000e+00**	0.000e+00	0.000e+00	3.333e+01	3.271e+01	4.353e+00	−	3.350e+01	3.279e+01	4.242e+00	−	3.373e+01	3.319e+01	4.141e+00	−
f_{10}	30	8.008e+01	8.023e+01	1.495e+01	**4.531e+01**	4.588e+01	1.263e+01	+	8.011e+01	7.937e+01	1.377e+01	=	**5.297e+01**	5.392e+01	1.203e+01	+
	50	2.183e+02	2.173e+02	2.000e+01	**1.871e+02**	1.863e+02	2.020e+01	+	2.974e+02	2.955e+02	1.961e+01	−	**1.765e+02**	1.749e+02	2.034e+01	+
f_{11}	30	**2.548e+01**	2.526e+01	1.854e+00	2.663e+01	2.533e+01	4.641e+00	−	2.761e+01	2.744e+01	1.658e+00	−	2.705e+01	2.684e+01	1.670e+00	−
	50	**5.263e+01**	5.268e+01	2.212e+00	5.751e+01	5.730e+01	1.923e+00	−	5.783e+01	5.784e+01	1.716e+00	−	5.686e+01	5.698e+01	1.847e+00	−
f_{12}	30	1.293e+04	1.324e+04	4.162e+03	**2.044e+03**	3.073e+03	2.817e+03	+	2.637e+04	2.540e+04	5.554e+03	−	**1.930e+03**	3.118e+03	3.229e+03	+
	50	8.996e+04	8.949e+04	2.001e+04	**1.051e+04**	1.280e+04	9.229e+03	+	1.894e+05	1.911e+05	3.140e+04	−	**7.859e+03**	1.079e+04	9.715e+03	+
f_{13}	30	**1.979e+00**	1.888e+00	3.977e−01	3.679e+00	3.785e+00	1.082e+00	−	4.396e+00	4.461e+00	5.141e−01	−	3.893e+00	4.088e+00	6.032e−01	−
	50	**7.093e+00**	7.074e+00	6.947e−01	1.276e+01	1.267e+01	1.451e+00	−	1.615e+01	1.600e+01	1.238e+00	−	1.320e+01	1.318e+01	1.332e+00	−
f_{14}	30	1.248e+01	1.248e+01	3.051e−01	1.254e+01	1.255e+01	3.082e−01	=	1.275e+01	1.271e+01	2.222e−01	−	**1.245e+01**	1.239e+01	3.126e−01	=
	50	**2.214e+01**	2.212e+01	2.642e−01	2.254e+01	2.250e+01	2.506e−01	−	2.263e+01	2.262e+01	1.905e−01	−	2.245e+01	2.242e+01	2.147e−01	−
f_{15}	30	**4.116e+01**	5.623e+01	5.212e+01	3.106e+02	3.471e+02	8.166e+01	−	4.000e+02	3.781e+02	7.639e+01	−	3.593e+02	3.628e+02	7.260e+01	−
	50	**1.301e+02**	1.422e+02	5.276e+01	4.000e+02	3.439e+02	8.621e+01	−	4.000e+02	3.801e+02	6.044e+01	−	4.000e+02	3.423e+02	9.073e+01	−
f_{16}	30	1.413e+02	1.453e+02	3.171e+01	**7.145e+01**	1.260e+02	1.226e+02	+	**1.036e+02**	1.032e+02	1.660e+01	+	**7.016e+01**	1.238e+02	1.270e+02	+
	50	1.967e+02	1.969e+02	3.751e+01	**1.344e+02**	1.427e+02	5.437e+01	+	2.127e+02	2.129e+02	1.412e+01	−	**1.262e+02**	1.398e+02	5.774e+01	+
f_{17}	30	2.084e+02	2.134e+02	3.624e+01	**1.074e+02**	1.503e+02	1.337e+02	+	**1.557e+02**	1.585e+02	4.209e+01	+	**1.093e+02**	2.001e+02	1.521e+02	+
	50	2.767e+02	2.822e+02	3.860e+01	**1.935e+02**	2.005e+02	3.869e+01	+	**2.708e+02**	2.716e+02	1.624e+01	=	**1.877e+02**	1.938e+02	3.733e+01	+
f_{18}	30	9.132e+02	8.993e+02	7.031e+01	9.199e+02	9.008e+02	4.830e+01	−	**9.074e+02**	9.078e+02	1.582e+00	+	9.238e+02	9.073e+02	4.805e+01	−
	50	9.430e+02	9.414e+02	1.894e+01	**9.333e+02**	9.313e+02	2.106e+01	+	**9.339e+02**	9.287e+02	2.658e+01	+	**9.354e+02**	9.274e+02	3.420e+01	+
f_{19}	30	9.140e+02	9.102e+02	1.853e+01	9.192e+02	8.921e+02	5.571e+01	−	**9.078e+02**	9.058e+02	1.536e+01	+	9.226e+02	9.176e+02	3.155e+01	−
	50	9.435e+02	9.418e+02	1.317e+01	**9.328e+02**	9.184e+02	9.196e+01	+	**9.353e+02**	9.325e+02	1.912e+01	+	**9.335e+02**	9.234e+02	3.782e+01	+
f_{20}	30	9.132e+02	9.119e+02	8.572e+00	9.226e+02	9.173e+02	3.076e+01	−	**9.085e+02**	9.066e+02	1.547e+01	+	9.273e+02	9.095e+02	4.903e+01	−
	50	9.430e+02	9.438e+02	4.918e+00	**9.327e+02**	9.247e+02	3.453e+01	+	**9.345e+02**	9.290e+02	2.660e+01	+	**9.349e+02**	9.345e+02	1.121e+01	+
f_{21}	30	5.000e+02	5.000e+02	0.000e+00	5.000e+02	5.520e+02	1.781e+02	−	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	6.622e+02	2.806e+02	−
	50	5.000e+02	5.000e+02	0.000e+00	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	5.000e+02	0.000e+00	=	5.000e+02	5.329e+02	1.384e+02	=
f_{22}	30	9.603e+02	9.609e+02	1.477e+01	**9.143e+02**	9.133e+02	1.587e+01	+	**9.191e+02**	9.181e+02	1.020e+01	+	**9.093e+02**	9.099e+02	1.552e+01	+
	50	9.912e+02	9.917e+02	7.240e+00	**9.614e+02**	9.619e+02	8.822e+00	+	9.935e+02	9.934e+02	7.254e+00	=	**9.596e+02**	9.572e+02	8.136e+00	+
f_{23}	30	5.000e+02	5.000e+02	0.000e+00	5.000e+02	6.110e+02	2.349e+02	−	5.000e+02	5.060e+02	4.243e+01	=	5.000e+02	5.988e+02	2.338e+02	−
	50	5.000e+02	5.000e+02	0.000e+00	5.000e+02	5.251e+02	1.087e+02	=	5.000e+02	5.000e+02	0.000e+00	=	5.000e+02	5.120e+02	5.938e+01	=

(*continued on next page*)

Table 9 (*continued*)

f_i	D	CLPSO			CLPSO:DE/rand/1				CLPSO:DE/rand/2				CLPSO:TDE/rand/1			
		Median	*Mean*	*St.D.*	*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
f_{24}	30	2.000e+02	2.000e+02	0.000e+00	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	2.000e+02	2.000e+02	4.950e−03	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
f_{25}	30	1.659e+03	1.659e+03	4.102e+00	**1.648e+03**	1.646e+03	8.901e+00	+	1.664e+03	1.664e+03	3.667e+00	−	**1.643e+03**	1.643e+03	8.630e+00	+
	50	1.701e+03	1.702e+03	2.610e+00	**1.697e+03**	1.696e+03	5.477e+00	+	1.709e+03	1.709e+03	2.369e+00	−	**1.692e+03**	1.691e+03	5.838e+00	+
					Total number of (+/−/=): (24/16/10)				Total number of (+/−/=): (16/19/15)				Total number of (+/−/=): (24/17/9)			

convergence dynamics strongly depend on the best personal positions of the swarm. Hence, it is expected that the evolutionary process of the proposed framework will have a great impact on their performance. This can be validated by the experimental results reported in Table 7 and Table 8. It is evident that the proposed framework significantly enhances the performance of both algorithms for almost all benchmark functions. Both algorithms exhibit inferior performance only in the multimodal f_8 function. Generally speaking, in PSO variants where the update rule strongly depends on the swarm's best personal positions (such as the FIPS and UPSO algorithms), an intelligent evolution of the best personal positions is highly recommended, since it may yield significant improvements in the quality of the solutions, with a relatively small computational overhead. Moreover, based on statistical rankings of their performance, all hybrids demonstrate statistical significant performance differences against their corresponding PSO algorithm, while the most promising hybrid for both algorithms is the one that incorporates the TDE/rand/1 mutation strategy, i.e., FIPS:TDE/rand/1, and UPSO:TDE/rand/1. The hybrid algorithms that employ the DE/rand/1 perform marginally worse that the corresponding hybrids with the TDE/rand/1 mutation strategy.

This subsection ends with a discussion on the experimental results of the proposed framework, applied to the CLPSO algorithm [52]. As demonstrated in Table 9, this is the only variant in which the proposed approach causes deteriorated performance in almost all multimodal and expanded functions. The hybrid variants exhibit an increased performance on the half functions of the considered benchmark suite, which are mainly unimodal and hybrid composition functions.

In more detail, all hybrids operate similarly or produce improved performance in all unimodal functions ($f_1 - f_5$). In the hybrid functions CLPSO:DE/rand/1 and CLPSO:TDE/rand/1 behave similarly, with an increased performance on six functions (f_{16}, f_{17}, and $f_{22} - f_{25}$). It is interesting to observe that although their performance is decreased on the 30–dimensional cases of $f_{18} - f_{20}$, the opposite happens on the 50–dimensional cases. CLPSO:DE/rand/2 demonstrates an increased performance against CLPSO only in three hybrid composition functions ($f_{17} - f_{20}$). For the multimodal functions both CLPSO:DE/rand/1 and CLPSO:TDE/rand/1 result in significant performance gains only in f_{10} and f_{12} functions. Nevertheless, we have performed the aforementioned statistical ranking tests, which suggested that there exists a significant performance difference among the considered algorithms, with the best performing algorithm to be the CLPSO:DE/rand/1 algorithm. Moreover, both CLPSO:DE/rand/1 and CLPSO:TDE/rand/1 are statistically better in comparison with the original CLPSO algorithm, while marginal performance difference can be observed between CLPSO:DE/rand/1 and CLPSO:TDE/rand/1, as well as between CLPSO and CLPSO:TDE/rand/1 algorithms.

Trying to rationalize the aforementioned behavior, we may concentrate on CLPSO characteristics along with Differential Evolution dynamics. CLPSO updates each particle's velocity by utilizing a novel learning scheme based on the swarm's best personal positions. Specifically, CLPSO uses the best positions of the other particles as exemplars to be learned from. This strategy is applied to each dimension of a particle, producing a scheme in which each particle may potentially learn from different exemplars per dimension. Hence, CLPSO permits the particles to have more exemplars to learn from and a potentially larger search space to fly over. CLPSO has exhibited great performance gains mostly on multimodal functions, while it was not the best choice for solving unimodal problems [52]. As previously described, explorative DE mutation strategies tend to rapidly spread the individuals on the vicinity of the problem's minima. Depending on the problem at hand, the combination of the aforementioned characteristics may lead to conflicting effects. On unimodal benchmark functions, it may result in a more exploitative behavior and enhance its performance, while on the multimodal functions it may rapidly spread the best personal positions to many optima and enhance its explorative characteristics. Although the latter may increase its performance, it may also quickly drive the best personal positions to a stagnation stage [50] and subsequently hinder the performance of the hybrid PSO variant.

5.3. Hybridization of PSO with state-of-the-art DE variants

The experimental results end with the utilization of popular DE algorithms into the aforementioned state-of-the-art PSO variants. To keep the article compact we will incorporate four DE algorithms into the BBPSO and CLPSO variants, namely the Self-Adaptive Control Parameters in DE algorithm (jDE) [7], the Adaptive Differential Evolution with optional external archive algorithm (JADE) [112,111], the Differential Evolution algorithm with Strategy Adaptation (SaDE) [86,85], and the Differential Evolution with Global and Local Neighborhoods (DEGL) [14,9].

To this end, Table 10 and Table 11 demonstrate the experimental results of the proposed framework on BBPSO and CLPSO respectively. It can be easily observed that, for almost all functions, the resulting schemes exhibit either superior or equal performance in comparison with the original PSO variants. The impact of the proposed framework is evident mostly in the unimodal and the hybrid composition functions, in which there are statistical significant performance differences. It has to be noted that the behavior of the proposed framework on CLPSO has totally changed. Its performance is greatly enhanced by the proposed framework, in particular with the incorporation of the DEGL algorithm (CLPSO:DEGL), which, as presented in the next section, is one of the best performing hybrids considered in this study.

In addition, among the four hybrid schemes the best performance is observed in hybrids which implement either JADE or DEGL algorithms. In the BBPSO case, both BBPSO:JADE and BBPSO:DEGL result in significant performance gains in 13 out of 50 functions, while BBPSO:jDE and BBPSO:SADE in 10 out of 50 functions. In the CLPSO case the most promising approach is CLPSO:DEGL, since it demonstrates substantial performance improvements in 20 out of 50 functions.

To conclude, the incorporation of the proposed framework in any PSO variant is straightforward and, as suggested by our extensive experimental and statistical analysis, results in significant performance gains in the majority of the considered cases, with a relatively small computational overhead. Several combination with popular DE variants can be implemented,

Table 10
Error values of the BBPSO algorithm and their corresponding hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions.

Algorithm	D	f_1				f_2				f_3				f_4			
		Median	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
BBPSO	30	0.000e+00	0.000e+00	0.000e+00		6.000e−03	9.260e−03	8.480e−03		1.243e+06	1.295e+06	5.728e+05		1.962e+03	2.307e+03	1.188e+03	
	50	0.000e+00	0.000e+00	0.000e+00		2.407e+02	2.886e+02	1.452e+02		3.693e+06	3.709e+06	9.352e+05		3.026e+04	2.965e+04	6.094e+03	
BBPSO:jDE	30	0.000e+00	0.000e+00	0.000e+00	=	7.500e−03	2.158e−02	4.149e−02	=	1.174e+06	1.249e+06	5.851e+05	=	1.479e+03	1.654e+03	9.787e+02	+
	50	0.000e+00	0.000e+00	0.000e+00	=	2.456e+02	2.779e+02	1.173e+02	=	3.624e+06	3.767e+06	1.067e+06	=	2.789e+04	2.911e+04	6.197e+03	=
BBPSO:JADE	30	0.000e+00	0.000e+00	0.000e+00	=	3.000e−03	1.166e−02	2.664e−02	+	1.314e+06	1.441e+06	6.787e+05	=	9.637e+02	1.330e+03	9.517e+02	+
	50	0.000e+00	0.000e+00	0.000e+00	=	2.527e+02	2.748e+02	1.226e+02	=	3.619e+06	3.815e+06	1.077e+06	=	2.622e+04	2.627e+04	5.887e+03	+
BBPSO:SADE	30	0.000e+00	0.000e+00	0.000e+00	=	6.000e−03	1.304e−02	2.119e−02	=	1.396e+06	1.472e+06	6.310e+05	=	9.130e+02	1.083e+03	6.935e+02	+
	50	0.000e+00	0.000e+00	0.000e+00	=	2.616e+02	2.803e+02	1.379e+02	=	4.127e+06	4.276e+06	1.360e+06	–	2.208e+04	2.257e+04	5.153e+03	+
BBPSO:DEGL	30	0.000e+00	0.000e+00	0.000e+00	=	1.300e−02	2.044e−02	1.992e−02	–	1.152e+06	1.244e+06	5.385e+05	=	8.792e+02	9.866e+02	6.383e+02	+
	50	0.000e+00	0.000e+00	0.000e+00	=	2.432e+02	2.753e+02	1.425e+02	=	3.702e+06	3.854e+06	1.284e+06	=	2.211e+04	2.302e+04	6.255e+03	+
		f_5				f_6				f_7				f_8			
BBPSO	30	5.243e+03	5.314e+03	1.033e+03		1.148e+01	2.796e+01	4.218e+01		4.696e+03	4.696e+03	5.039e−01		2.095e+01	2.094e+01	5.696e−02	
	50	1.295e+04	1.262e+04	1.969e+03		4.016e+01	5.831e+01	4.576e+01		6.195e+03	6.197e+03	4.553e+00		2.115e+01	2.114e+01	2.978e−02	
BBPSO:jDE	30	4.585e+03	4.733e+03	9.571e+02	+	1.802e+01	3.151e+01	3.791e+01	=	4.696e+03	4.696e+03	1.098e−02	=	2.096e+01	2.096e+01	4.627e−02	=
	50	1.224e+04	1.212e+04	1.641e+03	=	8.146e+01	7.447e+01	4.285e+01	=	6.195e+03	6.196e+03	1.758e+00	=	2.114e+01	2.114e+01	3.791e−02	=
BBPSO:JADE	30	4.726e+03	4.657e+03	7.968e+02	+	1.616e+01	3.926e+01	7.156e+01	=	4.696e+03	4.696e+03	1.414e−03	=	2.095e+01	2.095e+01	4.929e−02	=
	50	1.164e+04	1.162e+04	1.791e+03	+	7.508e+01	6.557e+01	3.765e+01	=	6.195e+03	6.196e+03	1.006e+00	+	2.113e+01	2.113e+01	4.247e−02	=
BBPSO:SADE	30	4.775e+03	4.770e+03	1.059e+03	+	1.887e+01	4.006e+01	4.486e+01	–	4.696e+03	4.696e+03	2.399e−03	=	2.097e+01	2.096e+01	6.451e−02	=
	50	1.166e+04	1.181e+04	1.643e+03	+	7.619e+01	7.304e+01	5.615e+01	=	6.195e+03	6.195e+03	2.709e−01	+	2.114e+01	2.113e+01	3.721e−02	=
BBPSO:DEGL	30	4.578e+03	4.480e+03	1.035e+03	+	2.040e+01	4.057e+01	4.745e+01	–	4.696e+03	4.696e+03	1.116e−01	=	2.097e+01	2.095e+01	5.331e−02	=
	50	1.069e+04	1.076e+04	1.942e+03	+	7.396e+01	7.125e+01	5.296e+01	=	6.195e+03	6.197e+03	4.057e+00	=	2.114e+01	2.114e+01	3.603e−02	=
		f_9				f_{10}				f_{11}				f_{12}			
BBPSO	30	5.622e+01	5.635e+01	1.079e+01		7.213e+01	7.556e+01	1.919e+01		2.790e+01	2.802e+01	1.926e+00		1.590e+03	2.585e+03	2.746e+03	
	50	1.338e+02	1.329e+02	2.131e+01		1.691e+02	1.755e+02	4.101e+01		5.483e+01	5.471e+01	3.821e+00		1.447e+04	1.505e+04	9.270e+03	
BBPSO:jDE	30	1.232e+01	1.335e+01	5.592e+00	+	7.539e+01	8.057e+01	1.834e+01	=	2.835e+01	2.790e+01	2.507e+00	=	1.721e+03	2.834e+03	3.284e+03	=
	50	3.159e+01	3.377e+01	8.744e+00	+	1.726e+02	1.754e+02	3.789e+01	=	5.558e+01	5.532e+01	3.098e+00	=	1.341e+04	1.480e+04	9.558e+03	=
BBPSO:JADE	30	2.836e+01	2.973e+01	8.438e+00	+	8.009e+01	8.211e+01	2.184e+01	=	2.801e+01	2.771e+01	2.939e+00	=	1.904e+03	2.754e+03	2.711e+03	=
	50	8.353e+01	7.963e+01	1.457e+01	+	1.512e+02	1.605e+02	3.897e+01	=	5.527e+01	5.516e+01	2.880e+00	=	1.447e+04	1.708e+04	9.857e+03	=
BBPSO:SADE	30	4.792e+00	5.704e+00	3.343e+00	+	6.965e+01	7.927e+01	2.813e+01	=	2.838e+01	2.804e+01	2.568e+00	=	2.045e+03	2.841e+03	2.754e+03	=
	50	1.240e+01	1.261e+01	4.482e+00	+	1.740e+02	1.709e+02	3.390e+01	=	5.556e+01	5.451e+01	3.592e+00	=	2.010e+04	1.935e+04	1.053e+04	–
BBPSO:DEGL	30	4.875e+01	5.096e+01	1.355e+01	+	7.213e+01	7.223e+01	2.067e+01	=	2.795e+01	2.771e+01	2.493e+00	=	3.136e+03	4.576e+03	4.325e+03	–
	50	1.254e+02	1.221e+02	1.952e+01	+	1.562e+02	1.593e+02	4.102e+01	+	5.505e+01	5.474e+01	2.999e+00	=	1.408e+04	1.571e+04	8.759e+03	=
		f_{13}				f_{14}				f_{15}				f_{16}			
BBPSO	30	6.252e+00	6.503e+00	1.855e+00		1.237e+01	1.229e+01	3.984e−01		3.033e+02	3.249e+02	8.537e+01		1.187e+02	1.356e+02	5.300e+01	
	50	1.763e+01	1.839e+01	4.973e+00		2.223e+01	2.209e+01	4.659e−01		2.511e+02	2.834e+02	8.431e+01		1.318e+02	1.408e+02	3.960e+01	
BBPSO:jDE	30	3.909e+00	3.964e+00	1.116e+00	+	1.229e+01	1.229e+01	3.157e−01	=	3.000e+02	2.831e+02	1.036e+02	=	1.214e+02	1.616e+02	9.723e+01	=
	50	1.059e+01	1.152e+01	3.881e+00	+	2.221e+01	2.214e+01	3.570e−01	=	2.346e+02	2.692e+02	7.953e+01	=	1.306e+02	1.373e+02	2.888e+01	=
BBPSO:JADE	30	4.971e+00	5.284e+00	1.664e+00	+	1.246e+01	1.239e+01	3.553e−01	=	3.000e+02	3.311e+02	8.566e+01	=	1.358e+02	1.711e+02	1.049e+02	=
	50	1.521e+01	1.532e+01	4.473e+00	+	2.209e+01	2.207e+01	3.390e−01	=	2.388e+02	2.869e+02	9.089e+01	=	1.304e+02	1.394e+02	5.168e+01	=
BBPSO:SADE	30	2.104e+00	2.249e+00	5.793e−01	+	1.239e+01	1.238e+01	4.176e−01	=	3.000e+02	2.891e+02	9.370e+01	=	1.140e+02	1.390e+02	7.860e+01	=
	50	5.190e+00	5.265e+00	1.102e+00	+	2.219e+01	2.210e+01	3.993e−01	=	2.071e+02	2.688e+02	8.621e+01	=	1.340e+02	1.405e+02	3.155e+01	=
BBPSO:DEGL	30	4.807e+00	4.755e+00	1.279e+00	+	1.248e+01	1.245e+01	3.008e−01	–	3.000e+02	3.244e+02	8.845e+01	=	1.251e+02	1.597e+02	9.832e+01	=
	50	1.328e+01	1.369e+01	4.105e+00	+	2.209e+01	2.205e+01	4.329e−01	=	3.452e+02	3.145e+02	8.870e+01	=	1.190e+02	1.289e+02	5.551e+01	+

		f_{17}				f_{18}				f_{19}				f_{20}			
BBPSO	30	1.580e+02	1.737e+02	6.599e+01		9.270e+02	9.101e+02	4.536e+01		9.247e+02	9.209e+02	3.222e+01		9.247e+02	9.167e+02	3.558e+01	
	50	2.038e+02	2.099e+02	4.542e+01		9.928e+02	9.935e+02	2.367e+01		9.866e+02	9.913e+02	1.812e+01		9.868e+02	9.890e+02	2.113e+01	
BBPSO:jDE	30	1.433e+02	1.640e+02	7.868e+01	=	9.249e+02	9.191e+02	3.129e+01	=	9.238e+02	9.156e+02	3.488e+01	=	9.231e+02	9.202e+02	2.576e+01	=
	50	2.037e+02	2.104e+02	4.645e+01	=	9.826e+02	9.835e+02	3.118e+01	+	9.930e+02	9.921e+02	1.728e+01	=	9.896e+02	9.860e+02	3.296e+01	=
BBPSO:JADE	30	1.410e+02	1.842e+02	9.689e+01	=	9.215e+02	9.068e+02	4.400e+01	+	9.265e+02	9.149e+02	3.935e+01	=	9.223e+02	9.095e+02	4.115e+01	=
	50	1.941e+02	2.004e+02	6.386e+01	=	9.774e+02	9.818e+02	1.820e+01	+	9.869e+02	9.872e+02	1.819e+01	=	9.910e+02	9.904e+02	1.693e+01	=
BBPSO:SADE	30	1.446e+02	1.616e+02	6.112e+01	=	9.266e+02	9.211e+02	3.201e+01	=	9.227e+02	9.128e+02	3.856e+01	=	9.249e+02	9.167e+02	3.540e+01	=
	50	2.101e+02	2.062e+02	4.960e+01	=	9.849e+02	9.879e+02	1.615e+01	=	9.815e+02	9.843e+02	1.978e+01	+	9.838e+02	9.804e+02	4.138e+01	=
BBPSO:DEGL	30	1.435e+02	1.789e+02	1.050e+02	=	9.228e+02	9.166e+02	3.037e+01	=	9.246e+02	9.135e+02	3.880e+01	=	9.227e+02	9.036e+02	4.970e+01	=
	50	1.859e+02	1.910e+02	5.182e+01	=	9.823e+02	9.840e+02	1.814e+01	+	9.851e+02	9.846e+02	1.684e+01	=	9.729e+02	9.713e+02	3.957e+01	+
		f_{21}				f_{22}				f_{23}				f_{24}			
BBPSO	30	5.000e+02	5.060e+02	4.243e+01		9.649e+02	9.675e+02	2.823e+01		5.000e+02	5.653e+02	1.887e+02		2.000e+02	2.000e+02	0.000e+00	
	50	5.000e+02	5.199e+02	1.063e+02		1.006e+03	1.006e+03	1.564e+01		5.000e+02	5.000e+02	0.000e+00		2.000e+02	2.211e+02	1.374e+02	
BBPSO:jDE	30	5.000e+02	5.263e+02	9.080e+01	=	9.486e+02	9.525e+02	2.552e+01	+	5.000e+02	5.510e+02	1.517e+02	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.000e+02	0.000e+00	=	1.004e+03	1.004e+03	1.626e+01	=	5.000e+02	5.272e+02	1.347e+02	=	2.000e+02	2.187e+02	1.325e+02	=
BBPSO:JADE	30	5.000e+02	5.252e+02	1.090e+02	=	9.488e+02	9.514e+02	2.365e+01	+	5.000e+02	5.592e+02	1.854e+02	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.060e+02	4.243e+01	=	1.010e+03	1.008e+03	1.336e+01	=	5.000e+02	5.262e+02	1.152e+02	=	2.000e+02	2.200e+02	1.413e+02	=
BBPSO:SADE	30	5.000e+02	5.507e+02	1.513e+02	=	9.671e+02	9.679e+02	2.659e+01	=	5.000e+02	5.386e+02	1.419e+02	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.062e+02	4.355e+01	=	1.006e+03	1.008e+03	1.539e+01	=	5.000e+02	5.334e+02	1.407e+02	=	2.000e+02	2.811e+02	2.779e+02	=
BBPSO:DEGL	30	5.000e+02	5.000e+02	0.000e+00	=	9.522e+02	9.559e+02	2.997e+01	+	5.000e+02	5.137e+02	9.722e+01	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.141e+02	9.977e+01	=	1.005e+03	1.003e+03	1.306e+01	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.603e+02	2.410e+02	=
		f_{25}															
BBPSO	30	1.669e+03	1.668e+03	7.609e+00													
	50	1.724e+03	1.724e+03	6.689e+00													
BBPSO:jDE	30	1.665e+03	1.666e+03	6.955e+00	=	Total number of (+/−/=): (9/0/41)											
	50	1.720e+03	1.720e+03	6.691e+00	+												
BBPSO:JADE	30	1.670e+03	1.669e+03	7.077e+00	=	Total number of (+/−/=): (13/0/37)											
	50	1.723e+03	1.723e+03	6.704e+00	=												
BBPSO:SADE	30	1.667e+03	1.668e+03	6.660e+00	=	Total number of (+/−/=): (10/3/37)											
	50	1.723e+03	1.724e+03	7.326e+00	=												
BBPSO:DEGL	30	1.667e+03	1.667e+03	6.809e+00	=	Total number of (+/−/=): (13/4/33)											
	50	1.722e+03	1.722e+03	7.218e+00	=												

Table 11
Error values of the CLPSO algorithm and their corresponding hybrid DE variants on the 30 and 50-dimensional CEC 2005 benchmark functions

Algorithm	D	f_1				f_2				f_3				f_4			
		Median	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*		*Median*	*Mean*	*St.D.*	
CLPSO	30	0.000e+00	0.000e+00	0.000e+00		3.828e+02	3.828e+02	1.060e+02		1.204e+07	1.188e+07	3.107e+06		5.481e+03	5.396e+03	1.250e+03	
	50	0.000e+00	0.000e+00	0.000e+00		1.013e+04	1.021e+04	1.357e+03		5.084e+07	4.930e+07	1.161e+07		3.477e+04	3.428e+04	5.637e+03	
CLPSO:jDE	30	0.000e+00	0.000e+00	0.000e+00	=	3.384e+02	3.316e+02	1.032e+02	+	1.145e+07	1.120e+07	3.205e+06	=	5.132e+03	5.106e+03	1.634e+03	=
	50	0.000e+00	0.000e+00	0.000e+00	=	9.389e+03	9.517e+03	1.574e+03	+	5.261e+07	5.196e+07	1.053e+07	=	3.425e+04	3.427e+04	5.186e+03	=
CLPSO:JADE	30	0.000e+00	0.000e+00	0.000e+00	=	1.884e+02	1.951e+02	6.106e+01	+	1.000e+07	9.867e+06	3.015e+06	+	4.194e+03	4.229e+03	1.055e+03	+
	50	0.000e+00	0.000e+00	0.000e+00	=	7.163e+03	7.267e+03	1.331e+03	+	3.768e+07	3.813e+07	8.497e+06	+	2.857e+04	2.859e+04	5.809e+03	+
CLPSO:SADE	30	0.000e+00	0.000e+00	0.000e+00	=	1.172e+02	1.365e+02	7.185e+01	+	1.188e+07	1.229e+07	3.305e+06	=	3.229e+03	3.256e+03	1.163e+03	+
	50	0.000e+00	0.000e+00	0.000e+00	=	3.652e+03	4.199e+03	1.631e+03	+	5.332e+07	5.227e+07	9.488e+06	=	2.347e+04	2.348e+04	4.820e+03	+
CLPSO:DEGL	30	0.000e+00	0.000e+00	0.000e+00	=	1.131e+02	1.144e+02	3.671e+01	+	5.214e+06	5.313e+06	1.518e+06	+	2.935e+03	3.101e+03	1.018e+03	+
	50	0.000e+00	0.000e+00	0.000e+00	=	6.699e+03	6.928e+03	1.277e+03	+	3.709e+07	3.689e+07	9.227e+06	+	2.745e+04	2.808e+04	4.855e+03	+
		f_5				f_6				f_7				f_8			
CLPSO	30	4.011e+03	4.001e+03	4.276e+02		7.369e+00	1.779e+01	2.285e+01		4.696e+03	4.696e+03	1.837e−12		2.072e+01	2.072e+01	5.905e−02	
	50	9.753e+03	9.698e+03	7.903e+02		8.998e+01	8.705e+01	3.757e+01		6.195e+03	6.195e+03	4.594e−12		2.105e+01	2.104e+01	4.617e−02	
CLPSO:jDE	30	3.829e+03	3.795e+03	4.181e+02	+	7.038e+00	1.744e+01	2.225e+01	=	4.696e+03	4.696e+03	1.837e−12	=	2.074e+01	2.072e+01	6.404e−02	=
	50	9.311e+03	9.268e+03	8.360e+02	+	1.038e+02	1.031e+02	4.247e+01	=	6.195e+03	6.195e+03	4.594e−12	=	2.106e+01	2.105e+01	5.017e−02	=
CLPSO:JADE	30	3.778e+03	3.836e+03	3.654e+02	+	1.060e+01	1.984e+01	2.327e+01	=	4.696e+03	4.696e+03	1.837e−12	=	2.075e+01	2.073e+01	7.820e−02	=
	50	9.124e+03	9.222e+03	9.158e+02	+	9.471e+01	9.106e+01	3.912e+01	=	6.195e+03	6.195e+03	4.594e−12	=	2.105e+01	2.105e+01	4.404e−02	=
CLPSO:SADE	30	3.983e+03	4.026e+03	4.631e+02	=	1.563e+01	2.283e+01	2.454e+01	=	4.696e+03	4.696e+03	1.837e−12	=	2.074e+01	2.074e+01	5.717e−02	=
	50	9.624e+03	9.619e+03	9.466e+02	=	9.110e+01	9.131e+01	3.783e+01	=	6.195e+03	6.195e+03	4.594e−12	=	2.105e+01	2.105e+01	4.747e−02	=
CLPSO:DEGL	30	3.650e+03	3.640e+03	4.758e+02	+	9.552e+00	1.912e+01	2.190e+01	=	4.696e+03	4.696e+03	1.837e−12	=	2.074e+01	2.074e+01	5.682e−02	=
	50	9.131e+03	9.085e+03	9.073e+02	+	9.070e+01	9.109e+01	3.637e+01	=	6.195e+03	6.195e+03	4.594e−12	=	2.107e+01	2.106e+01	3.639e−02	=
		f_9				f_{10}				f_{11}				f_{12}			
CLPSO	30	0.000e+00	0.000e+00	0.000e+00		8.008e+01	8.023e+01	1.495e+01		2.548e+01	2.526e+01	1.854e+00		1.293e+04	1.324e+04	4.162e+03	
	50	0.000e+00	0.000e+00	0.000e+00		2.183e+02	2.173e+02	2.000e+01		5.263e+01	5.268e+01	2.212e+00		8.996e+04	8.949e+04	2.001e+04	
CLPSO:jDE	30	0.000e+00	0.000e+00	0.000e+00	=	7.661e+01	7.760e+01	1.517e+01	=	2.576e+01	2.541e+01	2.171e+00	=	1.168e+04	1.320e+04	4.979e+03	=
	50	0.000e+00	0.000e+00	0.000e+00	=	2.190e+02	2.162e+02	2.560e+01	=	5.304e+01	5.286e+01	1.965e+00	=	8.834e+04	8.706e+04	1.916e+04	=
CLPSO:JADE	30	0.000e+00	1.990e−02	1.407e−01	=	7.634e+01	7.616e+01	1.184e+01	=	2.541e+01	2.529e+01	1.496e+00	=	1.226e+04	1.263e+04	4.373e+03	=
	50	0.000e+00	1.990e−02	1.407e−01	=	2.083e+02	2.111e+02	2.337e+01	=	5.277e+01	5.271e+01	1.916e+00	=	8.098e+04	8.215e+04	1.577e+04	=
CLPSO:SADE	30	0.000e+00	0.000e+00	0.000e+00	=	7.139e+01	7.468e+01	1.474e+01	=	2.577e+01	2.539e+01	1.523e+00	=	1.409e+04	1.324e+04	4.115e+03	=
	50	0.000e+00	0.000e+00	0.000e+00	=	2.176e+02	2.177e+02	2.397e+01	=	5.276e+01	5.283e+01	1.807e+00	=	7.727e+04	7.933e+04	1.722e+04	+
CLPSO:DEGL	30	0.000e+00	0.000e+00	0.000e+00	=	7.408e+01	7.355e+01	1.145e+01	+	2.514e+01	2.502e+01	1.380e+00	=	1.298e+04	1.333e+04	4.428e+03	=
	50	0.000e+00	1.990e−02	1.407e−01	=	2.022e+02	2.010e+02	2.098e+01	+	5.331e+01	5.298e+01	2.258e+00	=	9.130e+04	9.142e+04	1.726e+04	=
		f_{13}				f_{14}				f_{15}				f_{16}			
CLPSO	30	1.979e+00	1.888e+00	3.977e−01		1.248e+01	1.248e+01	3.051e−01		4.116e+01	5.623e+01	5.212e+01		1.413e+02	1.453e+02	3.171e+01	
	50	7.093e+00	7.074e+00	6.947e−01		2.214e+01	2.212e+01	2.642e−01		1.301e+02	1.422e+02	5.276e+01		1.967e+02	1.969e+02	3.751e+01	
CLPSO:jDE	30	2.085e+00	2.000e+00	3.761e−01	=	1.257e+01	1.250e+01	3.046e−01	=	2.857e+01	4.064e+01	4.865e+01	+	1.393e+02	1.440e+02	3.052e+01	=
	50	6.852e+00	6.995e+00	7.806e−01	=	2.223e+01	2.218e+01	3.182e−01	=	1.096e+02	1.228e+02	5.394e+01	=	1.948e+02	1.998e+02	3.684e+01	=
CLPSO:JADE	30	2.131e+00	2.107e+00	2.973e−01	−	1.255e+01	1.252e+01	2.482e−01	=	5.075e+01	7.378e+01	6.405e+01	=	1.237e+02	1.293e+02	3.214e+01	+
	50	7.303e+00	7.290e+00	6.558e−01	=	2.223e+01	2.218e+01	2.394e−01	=	1.337e+02	1.357e+02	4.519e+01	=	1.798e+02	1.811e+02	3.379e+01	+
CLPSO:SADE	30	1.809e+00	1.803e+00	3.047e−01	=	1.253e+01	1.248e+01	3.113e−01	=	5.907e+01	8.021e+01	7.144e+01	=	1.407e+02	1.455e+02	3.519e+01	=
	50	4.431e+00	4.510e+00	6.041e−01	+	2.232e+01	2.223e+01	2.663e−01	−	1.355e+02	1.368e+02	5.355e+01	=	1.907e+02	1.996e+02	3.631e+01	=
CLPSO:DEGL	30	2.057e+00	2.043e+00	2.510e−01	=	1.258e+01	1.259e+01	2.758e−01	=	4.277e+01	6.249e+01	5.778e+01	=	1.272e+02	1.292e+02	2.906e+01	+
	50	7.037e+00	7.078e+00	7.025e−01	=	2.212e+01	2.209e+01	3.147e−01	=	1.227e+02	1.390e+02	4.285e+01	=	1.759e+02	1.792e+02	3.028e+01	+

		f_{17}				f_{18}				f_{19}				f_{20}			
CLPSO	30	2.084e+02	2.134e+02	3.624e+01		9.132e+02	8.993e+02	7.031e+01		9.140e+02	9.102e+02	1.853e+01		9.132e+02	9.119e+02	8.572e+00	
	50	2.767e+02	2.822e+02	3.860e+01		9.430e+02	9.414e+02	1.894e+01		9.435e+02	9.418e+02	1.317e+01		9.430e+02	9.438e+02	4.918e+00	
CLPSO:jDE	30	1.907e+02	1.924e+02	4.072e+01	+	9.138e+02	9.113e+02	1.610e+01	=	9.133e+02	9.092e+02	2.092e+01	=	9.133e+02	8.957e+02	4.114e+01	=
	50	2.674e+02	2.718e+02	3.792e+01	=	9.408e+02	9.416e+02	5.919e+00	+	9.410e+02	9.415e+02	5.000e+00	=	9.411e+02	9.418e+02	4.779e+00	+
CLPSO:JADE	30	1.932e+02	1.938e+02	4.534e+01	+	9.123e+02	9.067e+02	2.437e+01	=	9.133e+02	9.067e+02	2.615e+01	=	9.134e+02	9.048e+02	2.968e+01	=
	50	2.625e+02	2.644e+02	3.457e+01	+	9.428e+02	9.430e+02	6.845e+00	=	9.406e+02	9.418e+02	5.408e+00	=	9.415e+02	9.423e+02	4.524e+00	=
CLPSO:SADE	30	2.041e+02	2.086e+02	4.078e+01	=	9.135e+02	9.113e+02	1.598e+01	=	9.132e+02	9.094e+02	2.016e+01	=	9.129e+02	9.089e+02	2.073e+01	=
	50	2.631e+02	2.655e+02	3.555e+01	+	9.425e+02	9.421e+02	4.274e+00	=	9.419e+02	9.431e+02	5.518e+00	=	9.418e+02	9.417e+02	3.588e+00	+
CLPSO:DEGL	30	1.826e+02	1.911e+02	4.342e+01	+	9.122e+02	9.057e+02	2.632e+01	+	9.130e+02	9.041e+02	3.018e+01	+	9.126e+02	9.063e+02	2.605e+01	+
	50	2.646e+02	2.649e+02	3.706e+01	+	9.412e+02	9.396e+02	1.669e+01	=	9.420e+02	9.424e+02	5.561e+00	=	9.407e+02	9.413e+02	4.348e+00	+
		f_{21}				f_{22}				f_{23}				f_{24}			
CLPSO	30	5.000e+02	5.000e+02	0.000e+00		9.603e+02	9.609e+02	1.477e+01		5.000e+02	5.000e+02	0.000e+00		2.000e+02	2.000e+02	0.000e+00	
	50	5.000e+02	5.000e+02	0.000e+00		9.912e+02	9.917e+02	7.240e+00		5.000e+02	5.000e+02	0.000e+00		2.000e+02	2.000e+02	4.950e−03	
CLPSO:jDE	30	5.000e+02	5.000e+02	0.000e+00	=	9.572e+02	9.591e+02	1.301e+01	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.000e+02	0.000e+00	=	9.868e+02	9.860e+02	8.951e+00	+	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	1.414e−04	=
CLPSO:JADE	30	5.000e+02	4.990e+02	7.228e+00	=	9.600e+02	9.590e+02	1.420e+01	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.000e+02	0.000e+00	=	9.875e+02	9.881e+02	8.608e+00	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
CLPSO:SADE	30	5.000e+02	5.000e+02	0.000e+00	=	9.613e+02	9.600e+02	1.627e+01	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.000e+02	0.000e+00	=	9.897e+02	9.888e+02	8.623e+00	=	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
CLPSO:DEGL	30	5.000e+02	5.000e+02	0.000e+00	=	9.522e+02	9.504e+02	1.464e+01	+	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
	50	5.000e+02	5.000e+02	0.000e+00	=	9.833e+02	9.817e+02	9.876e+00	+	5.000e+02	5.000e+02	0.000e+00	=	2.000e+02	2.000e+02	0.000e+00	=
		f_{25}															
CLPSO	30	1.659e+03	1.659e+03	4.102e+00													
	50	1.701e+03	1.702e+03	2.610e+00													
CLPSO:jDE	30	1.661e+03	1.660e+03	3.624e+00	=	Total number of (+/−/=): (9/0/41)											
	50	1.703e+03	1.702e+03	2.502e+00	=												
CLPSO:JADE	30	1.661e+03	1.661e+03	3.167e+00	−	Total number of (+/−/=): (12/2/36)											
	50	1.702e+03	1.702e+03	2.293e+00	=												
CLPSO:SADE	30	1.660e+03	1.660e+03	3.905e+00	=	Total number of (+/−/=): (8/1/41)											
	50	1.702e+03	1.702e+03	3.336e+00	=												
CLPSO:DEGL	30	1.660e+03	1.660e+03	3.292e+00	=	Total number of (+/−/=): (20/0/30)											
	50	1.702e+03	1.701e+03	2.490e+00	=												

in an attempt to produce an algorithm with increased performance, on a specific class of problems. Nevertheless, before implementing the proposed framework, one should pay attention on the behavior and the main characteristics of the applied algorithms. In the cases of PSO/DE hybridization one should consider the discussion in Section 4.2 and Section 4.3, along with other articles that discuss the characteristics of PSO and DE algorithms, e.g. [103,63,104,91,58,83,94,41,42,34,50]. In rare cases, the hybridization of a PSO variant with a DE mutation strategy may alter its convergence dynamics, which may lead to performance deterioration, e.g. the incorporation of an explorative DE mutation strategy in the CLPSO algorithm (refer to the last paragraphs of Section 5.2).

5.4. Overall performance

The objective of this work is to investigate whether given a well performing PSO algorithm, a performance improvement is possible through the proposed framework, without destroying its dynamics. Based on the aforementioned results, we can safely conclude that this objective has been successfully accomplished by the proposed framework for the majority of the considered algorithms. In this subsection, we conclude the presentation of the experimental results, by providing a summarizing comparison of all PSO variants over all benchmark functions. To this end, we include a statistical analysis for ranking the performance of all algorithms on all benchmark functions. We also visualize their behavior by firstly employing the Empirical Cumulative probability Distribution Function of the error as an overall performance procedure and afterwards by illustrating the convergence graphs of the median solution error values over several functions.

Table 12 demonstrates the average rankings computed through the Friedman, the Aligned Friedman, and the Quade tests. For each test we present the algorithm and its score in ascending order, while at the bottom of the table we illustrate the computed statistics and the corresponding p-values. Based on the reported p-values, all tests strongly suggests the existence of significant differences among the considered algorithms, at the $\alpha = 0.05$ level of significance. In general, it can be observed that in almost every test, the proposed hybrid variants occupy the first positions as well as they are better than their corresponding original PSO algorithm. In more detail, in the Friedman test, the first six positions are acquired by the six hybrid variants. Based on Friedman's characteristics the ranking indicates that the hybrid variants reach the first positions for the majority of the considered functions. Subsequently a similar behavior can be observed for the Aligned Friedman test. In this test, BBPSO reaches the fourth position, while the other variants follow a similar ranking order. Finally, in the Quade test the first place is acquired by the CLPSO:DEGL, the second by the CLPSO algorithm and the following seven positions by the proposed hybrid variants. This behavior indicates that CLPSO:DEGL, FIPS:TDE/rand/1 and CLPSO algorithms perform better on the most difficult benchmarks of the considered set. Based on the aforementioned tests, we can safely conclude that CLPSO:DEGL, CLPSO:DE/rand/1 and BBPSO/DE/rand/1 can be characterized as the best performing algorithms of the considered study.

Consequently, Fig. 7 illustrates the Empirical Cumulative probability Distribution Function (*ECDF*) of the solution error values for the PSO variants on all considered benchmark functions. As it can be clearly observed the hybrid PSO variants exhibit a great potential on the CEC 2005 function set. Almost all hybrid PSO variants have higher ECDF values compared with their original PSO algorithms. In general, the proposed hybrids produce one to two orders of magnitude less error values than the corresponding original PSO algorithm, e.g. the CLPSO:DE:rand/1, UPSO:TDE/rand/1, FIPS:TDE/rand/1 and BBPSO:DE/rand/1 curves reach unity at approximately 10^6 error, while the curves corresponding to the original PSO algorithms (CLPSO, UPSO, FIPS and BBPSO) at error 10^7, 10^8, 10^8, and 10^7, respectively (please refer to the zoomed subfigure inside Fig. 7).

Table 12
Average Rankings of all PSO variants over all benchmark functions

Average Ranking						
Rank	Friedman		Aligned Friedman		Quade	
	Algorithm	Score	Algorithm	Score	Algorithm	Score
1	CLPSO:DE/rand/1	5.329	CLPSO:DE/rand/1	243.590	CLPSO:DEGL	5.581
2	BBPSO:DE/rand/1	5.550	BBPSO:DE/rand/1	263.050	CLPSO	5.927
3	DMSPSO:DE/rand/1	5.889	BBPSO:JADE	290.519	FIPS:TDE/rand/1	5.982
4	FIPS:TDE/rand/1	5.979	BBPSO	299.540	DMSPSO:DE/rand/1	6.145
5	CLPSO:DEGL	6.329	DMSPSO:DE/rand/1	303.010	CLPSO:DE/rand/1	6.262
6	χPSO:DE/rand/1	6.440	CLPSO:DEGL	308.250	BBPSO:DE/rand/1	6.334
7	CLPSO	6.770	χPSO:DE/rand/1	314.879	χPSO:DE/rand/1	6.554
8	UPSO:TDE/rand/1	6.859	UPSO:TDE/rand/1	317.840	BBPSO:JADE	7.150
9	BBPSO:JADE	6.899	FIPS:TDE/rand/1	318.400	UPSO:TDE/rand/1	7.432
10	BBPSO	7.479	CLPSO	341.470	DMSPSO	7.893
11	DMSPSO	8.400	DMSPSO	374.679	BBPSO	7.961
12	χPSO	8.930	χPSO	400.329	χPSO	8.753
13	UPSO	11.339	UPSO	532.160	UPSO	10.842
14	FIPS	12.799	FIPS	599.279	FIPS	12.176
Statistic	179.7137		44.5192		10.4220	
p-value	1.2529e−10		2.5164e−5		3.4296e−20	

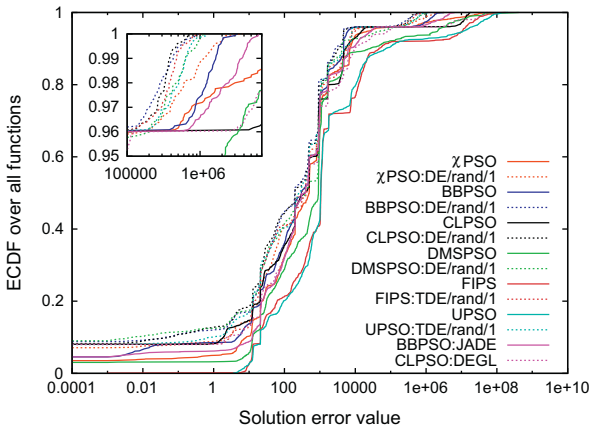

Fig. 7. Empirical cumulative probability distribution of the solution error for the state of the art PSO variants against the corresponding hybrid frameworks on the CEC 2005 benchmark functions.

Finally, in Fig. 8, we provide convergence graphs for 15 out of the 50-dimensional CEC 2005 benchmark functions, i.e. the $f_1 - f_6, f_9 - f_{13}, f_{16}, f_{17}, f_{21}$, and f_{23}. The graphs illustrate median solution error value curves for all PSO variants and their corresponding best performing hybrids considered in this work, obtained from 100 independent simulations. As expected, the graphs capture the previously observed behavior of the PSO algorithms, while they indicate that in most of the cases the

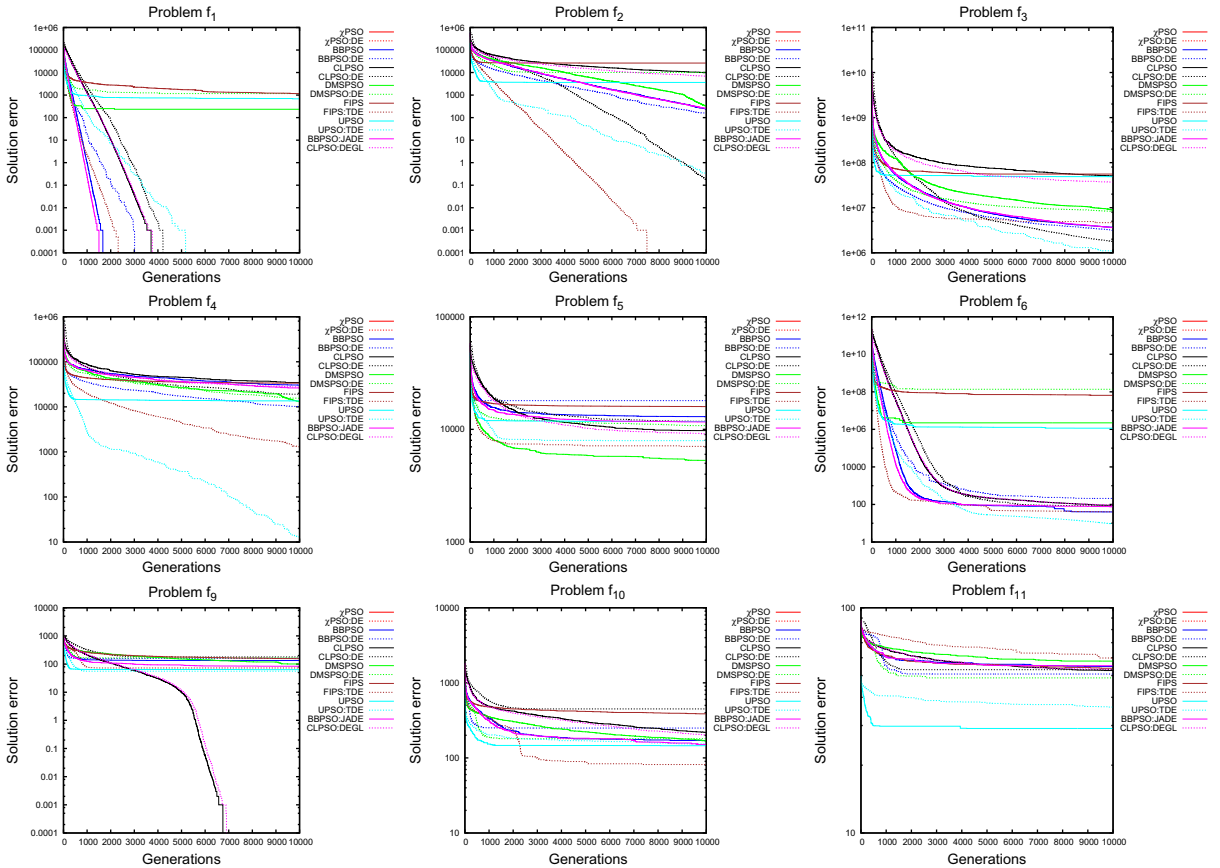

Fig. 8. Convergence graph (median curves) for all PSO variants and their corresponding best performing hybrids on the 50-dimensional $f_1 - f_6, f_9 - f_{13}, f_{16}, f_{17}, f_{21}$, and f_{23} CEC 2005 benchmark functions. The horizontal axis illustrates the number of generations, and the vertical axis illustrates the median of solution error values of 100 independent simulations.

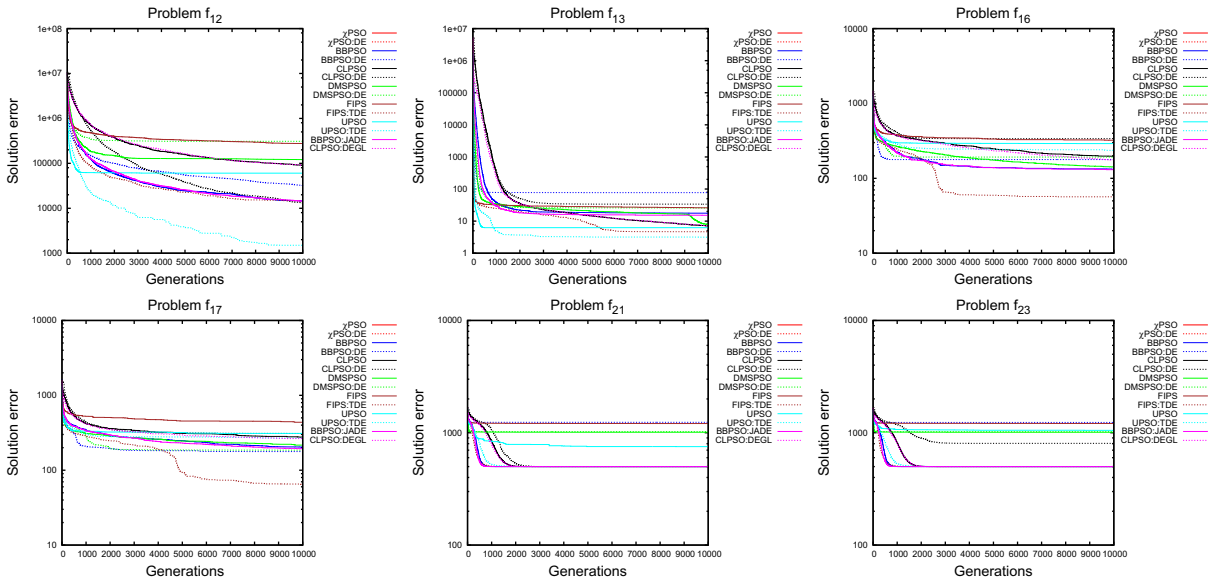

Fig. 8 (*continued*)

proposed framework either enhances the convergence of a PSO variant or operates similarly. There are relatively few cases where the proposed hybrids exhibit performance deterioration.

6. Concluding remarks

It has been recognized that during the evolutionary process of the Particle Swarm Optimization (PSO) algorithm the social and cognitive experience of each particle, i.e. the *memory-swarm*, tend to be distributed in the vicinity of the problem's optima. In this work, we attempt to take advantage of this characteristic behavior in an attempt to improve the performance of the algorithm. To this end, we propose a hybrid evolutionary framework to evolve the social and cognitive experience of the swarm and enhance the convergence properties of the PSO algorithm, without sacrificing its search capabilities. Here, we propose to incorporate the Differential Evolution algorithm, which is a simple and compact evolutionary algorithm exhibiting good convergence characteristics.

In detail, after each evolution step of the PSO algorithm, we evolve both social and cognitive experience of a swarm with the Differential Evolution algorithm. The newly evolved positions will be the outcome of the best experience of the swarm, which will have the potential to either locate better regions around problem's minima or to rapidly exploit already found promising regions, and thus accelerate convergence. The proposed hybridization framework is generic and scalable, as it is independent of the main evolution stages of any Particle Swarm Optimization variant. In this study we have applied it to the canonical PSO algorithm and five state-of-the-art PSO variants, while we have considered six classic DE mutation strategies and four popular DE variants.

Extensive experimental results on the CEC 2005 benchmark function suite validated by statistical significance analysis, demonstrate that the proposed framework is very promising. We have firstly employed in canonical PSO six DE mutation strategies separated in two groups, namely the explorative and the exploitative mutation strategies. An extensive experimental analysis clearly suggests that the explorative DE mutation strategies exhibited significant performance gains on the canonical PSO, over the majority of the considered benchmark functions. There are relatively few cases where the proposed framework results in performance deterioration. On the other hand, the performance improvement of the hybrids with exploitative mutation strategies was less prominent. We have observed that the more exploitative a mutation strategy is, the more it's performance deteriorates. All previously mentioned observations have been validated by a statistical analysis, including statistical significance tests and ranking procedures.

Based on the aforementioned analysis, we have further applied the three best performing DE mutation strategies to six state-of-the-art PSO variants. Extensive experimental analysis indicates that the incorporation of the proposed framework in any PSO variant was straightforward and significantly enhances the performance of the majority of the considered cases. The experimental analysis ended with the incorporation of four popular DE algorithms on the aforementioned PSO variants. Again, the resulting schemes demonstrated the expected behavior. Thus, the incorporation of the proposed framework in a PSO variant is highly recommended. Nevertheless, before implementing the proposed framework, one should pay attention on the behavior and the characteristics of the applied algorithms. In rare cases, the hybridization

of a PSO variant with a DE mutation strategy may alter its convergence dynamics, which may lead to performance deterioration.

The proposed framework can be easily extended in several different ways. Thereby, in a future work we intend to incorporate specialized mutation strategies with different characteristics [27] into the proposed framework, since their good explorative/exploitative capabilities may result in very competitive optimization algorithms. Furthermore, employing other optimization algorithms to the proposed framework, such as Evolutionary Strategies or Estimation of Distribution Algorithms, may also result in high potential hybrid PSO variants.

The behavior and the spatial characteristics of the swarm may result in great benefits in multimodal optimization. Thus, we intend to further study the swarm's spatial characteristics and its impact on locating many global optima [26], either in one or many structured swarms. Finally, we are interested in applying the proposed framework to large-scale and real-world optimization problems.

Acknowledgements

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. The authors thank the Associate Editor, and the anonymous reviewers for their valuable comments and suggestions that helped to improve the content as well as the clarity of the manuscript.

References

[1] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, in: EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII, Springer-Verlag, London, UK, 1998, pp. 601–610.
[2] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[3] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[4] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Transactions on Evolutionary Computation 10 (4) (2006) 459–472.
[5] T.M. Blackwell, P. Bentley, Improvised music with swarms, in: IEEE Congress on Evolutionary Computation, 2002, (CEC 2002), vol. 2, 2002, pp. 1462–1467.
[6] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: IEEE Swarm Intelligence Symposium, 2007, SIS 2007, IEEE, 2007, pp. 120–127.
[7] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-Adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation 10 (6) (2006) 646–657.
[8] U.K. Chakraborty, Advances in Differential Evolution. Springer Publishing Company, Incorporated, 2008.
[9] U.K. Chakraborty, S. Das, A. Konar, Differential evolution with local neighborhood, in: IEEE Congress on Evolutionary Computation, 2006, CEC 2006, 2006, pp. 2042–2049..
[10] M. Clerc, Particle Swarm Optimization, ISTE Publishing Company, 2006.
[11] M. Clerc, Standard Particle Swarm Optimization, from 2006 to 2011. Technical Report, <http://www.particleswarm.info/>, 2011.
[12] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[13] W.J. Conover, Practical Nonparametric Statistics, third ed., Wiley, 1999.
[14] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.
[15] S. Das, A. Abraham, A. Konar, Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives, in: Advances of Computational Intelligence in Industrial Systems, Studies in Computational Intelligence, vol. 116, Springer-Verlag, 2008, pp. 1–38..
[16] S. Das, A. Konar, U.K. Chakraborty, Improving particle swarm optimization with differentially perturbed velocity, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO '05, ACM, New York, NY, USA, 2005, pp. 177–184.
[17] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[18] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.
[19] R. Eberhart, P. Simpson, R. Dobbins, Computational Intelligence PC Tools, Academic Press Professional, Inc., San Diego, CA, USA, 1996.
[20] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings 6th Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43..
[21] A. Engelbrecht, Computational Intelligence: An Introduction, Halsted Press, 2002.
[22] A.P. Engelbrecht, Heterogeneous particle swarm optimization, in: Proceedings of the 7th International Conference on Swarm Intelligence, ANTS'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 191–202.
[23] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Balancing the exploration and exploitation capabilities of the differential evolution algorithm, in: IEEE Congress on Evolutionary Computation, 2008 (CEC 2008), IEEE World Congress on Computational Intelligence, 2008, pp. 2686–2693.
[24] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolutionary adaptation of the differential evolution control parameters, in: IEEE Congress on Evolutionary Computation, 2009 (CEC 2009), 2009, pp. 1359–1366.
[25] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential in: IEEE Congress on Evolutionary Computation, 2010 (CEC 2010), 2010, pp. 1–8.
[26] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Finding multiple global optima exploiting differential evolution's niching capability, in: IEEE Symposium on Differential Evolution (SDE), 2011, Paris, France, 2011, pp. 1–8.
[27] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing differential evolution utilizing proximity-based mutation operators, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 99–119.
[28] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Tracking particle swarm optimizers: an adaptive approach through multinomial distribution tracking with exponential forgetting, in: IEEE Congress on Evolutionary Computation, 2012 (CEC 2012), in press.
[29] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, Journal of Global Optimization 27 (2003) 105–129.

[30] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.
[31] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Information Sciences 181 (18) (2011) 3749–3765.
[32] S. Ghosh, S. Das, D. Kundu, K. Suresh, A. Abraham, Inter-particle communication and search-dynamics of lbest particle swarm optimizers: An analysis, Information Sciences 182 (1) (2012) 156–168.
[33] N. Hansen, Adaptive encoding: How to render search coordinate system invariant, in: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature: PPSN X, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 205–214.
[34] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, Impacts of invariance in search: when cma-es and pso face ill-conditioned and non-separable problems, Applied Soft Computing 11 (8) (2011) 5755–5769.
[35] Z. Hao, G. Guo, H. Huang, A particle swarm optimization algorithm with differential evolution, in: International Conference on Machine Learning and Cybernetics, 2007, vol. 2, 2007, pp. 1031–1035..
[36] W. Hart, N. Krasnogor, J. Smith, Memetic evolutionary algorithms, in: W. Hart, J. Smith, N. Krasnogor (Eds.), Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, vol. 166, Springer, Berlin/Heidelberg, 2005, pp. 3–27.
[37] T. Hendtlass, A combined swarm differential evolution algorithm for optimization problems, in: L. Monostori, J. Váncza, M. Ali (Eds.), Engineering of Intelligent Systems, Lecture Notes in Computer Science, vol. 2070, Springer, Berlin/Heidelberg, 2001, pp. 11–18.
[38] L.M. Hiot, Y.S. Ong, B.K. Panigrahi, Y. Shi, M.H. Lim (Eds.), Handbook of Swarm Intelligence, Adaptation, Learning, and Optimization, vol. 8, Springer, Berlin/Heidelberg, 2010.
[39] B. Hopkins, J.G. Skellam, A new method for determining the type of distribution of plant individuals, Annals of Botany 18 (2) (1954) 213–227.
[40] G. Iacca, F. Neri, E. Mininno, Y.S. Ong, M.H. Lim, Ockham's razor in memetic computing: three stage optimal memetic exploration, Information Sciences 188 (0) (2012) 17–43.
[41] A. Iorio, X. Li, Improving the performance and scalability of differential evolution, in: X. Li, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, Z. Michalewicz, T. Hendtlass, K. Deb, K. Tan, J. Branke, Y. Shi (Eds.), Simulated Evolution and Learning, Lecture Notes in Computer Science, vol. 5361, Springer, Berlin/Heidelberg, 2008, pp. 131–140.
[42] A. Iorio, X. Li, Improving the performance and scalability of differential evolution on problems exhibiting parameter interactions, journal Soft Computing – A Fusion of Foundations, Methodologies and Applications 15 (2011) 1769–1792.
[43] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2) (2012) 482–500.
[44] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
[45] S. Kannan, S.M.R. Slochanal, P. Subbaraj, N.P. Padhy, Application of particle swarm optimization technique and its variants to generation expansion planning problem, Electric Power Systems Research 70 (3) (2004) 203–210.
[46] J. Kennedy, Bare bones particle swarms, in: IEEE Swarm Intelligence Symposium, 2003, SIS '03, 2003, pp. 80–87.
[47] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings IEEE International Conference on Neural Networks, vol. 4, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[48] T. Krasnogor, Towards robust memetic algorithms, in: W. Hart, J. Smith, N. Krasnogor (Eds.), Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, vol. 166, Springer, Berlin/Heidelberg, 2005, pp. 185–207.
[49] T. Krink, M. Lovbjerg, The LifeCycle model: combining particle swarm optimisation, genetic algorithms and HillClimbers, in: J. Guervós, P. Adamidis, H.G. Beyer, H.P. Schwefel, J.L. Fernández-Villacañas (Eds.), Parallel Problem Solving from Nature PPSN VII, Lecture Notes in Computer Science, vol. 2439, Springer Berlin/Heidelberg, 2002, pp. 621–630.
[50] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing. <http://citeseer.ist.psu.edu/317991.html>, 2000, pp. 76–83.
[51] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation 16 (2) (2012) 210–224.
[52] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.
[53] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of the 2005 IEEE Swarm Intelligence Symposium, 2005, SIS '05, 2005, pp. 124–129.
[54] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Applied Soft Computing 10 (2) (2010) 629–640.
[55] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing – A Fusion of Foundations, Methodologies and Applications 9 (6) (2005) 448–462.
[56] R. Mendes, Population Topologies and Their Influence in Particle Swarm Performance. Ph.D. thesis; Escola de Engenharia, Universidade do Minho, 1977.
[57] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (2004) 204–210.
[58] C.K. Monson, K.D. Seppi, Exposing origin-seeking bias in PSO, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO '05, ACM, New York, NY, USA, 2005, pp. 241–248
[59] M.A. Montes de Oca, C. Cotta, F. Neri, Local search, in: F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, vol. 379, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 29–41.
[60] M.A. Montes de Oca, J. Peña, T. Stützle, C. Pinciroli, M. Dorigo, Heterogeneous particle swarm optimizers, in: P. Haddow, et al. (Eds.), IEEE Congress on Evolutionary Computation, 2009 (CEC 2009), IEEE Press, Piscataway, NJ, 2009, pp. 698–705.
[61] M.A. Montes de Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1120–1132.
[62] F. Neri, C. Cotta, A primer on memetic algorithms, in: F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, Springer, Berlin, Heidelberg, 2012, pp. 43–52.
[63] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Information Sciences 181 (12) (2011) 2469–2487.
[64] F. Neri, V. Tirronen, Scale factor local search in differential evolution, Memetic Computing 1 (2009) 153–171.
[65] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artificial Intelligence Review 33 (2010) 61–106.
[66] O. Olorunda, A.P. Engelbrecht, An analysis of heterogeneous cooperative algorithms, in: IEEE Congress on Evolutionary Computation, 2009 (CEC 2009), 2009, pp. 1562–1569.
[67] M. Omran, A. Salman, A. Engelbrecht, Self-adaptive differential evolution, in: Y. Hao, J. Liu, Y. Wang, Y.M. Cheung, H. Yin, L. Jiao, J. Ma, Y.C. Jiao (Eds.), Computational Intelligence and Security, Lecture Notes in Computer Science, vol. 3801, Springer, Berlin/Heidelberg, 2005, pp. 192–199.
[68] M.G.H. Omran, A.P. Engelbrecht, A. Salman, Differential evolution based particle swarm optimization, in: IEEE Swarm Intelligence Symposium 2007, SIS 2007, 2007, pp. 112–119.
[69] Y.S. Ong, M. Lim, X. Chen, Memetic computation – past, present – future research frontier, IEEE Computational Intelligence Magazine 5 (2) (2010) 24–31.
[70] M. Pant, R. Thangaraj, A. Abraham, DE-PSO: a new hybrid meta-heuristic for solving global optimization problems, New Mathematics and Natural Computation 07 (03) (2011) 363.

[71] M. Pant, R. Thangaraj, C. Grosan, A. Abraham, Hybrid differential evolution – particle swarm optimization algorithm for solving global optimization problems, in: Third International Conference on Digital Information Management, 2008, ICDIM 2008, 2008, pp. 18–24..
[72] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Stretching technique for obtaining global minimizers through particle swarm optimization, in: Particle Swarm Optimization Workshop, Indianapolis, USA, 2001, pp. 22–29.
[73] K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1 (2-3) (2002) 235–306.
[74] K.E. Parsopoulos, M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 211–224.
[75] K.E. Parsopoulos, M.N. Vrahatis, UPSO: a unified particle swarm optimization scheme, in: Lecture Series on Computer and Computational Sciences, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), vol. 1, 2004, pp. 868–873.
[76] K.E. Parsopoulos, M.N. Vrahatis, Parameter selection and adaptation in unified particle swarm optimization, Mathematical and Computer Modelling 46 (1-2) (2007) 198–213.
[77] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference (IGI Global), Hershey, PA, USA, 2010.
[78] N.G. Pavlidis, D.K. Tasoulis, V.P. Plagianakos, M.N. Vrahatis, Human designed vs. genetically programmed differential evolution operators, in: IEEE Congress on Evolutionary Computation, 2006 (CEC 2006), 2006, pp. 1880–1886.
[79] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in: IEEE Swarm Intelligence Symposium 2003, SIS '03, 2003, pp. 174–181.
[80] V.P. Plagianakos, M.N. Vrahatis, Parallel evolutionary training algorithms for 'hardware–friendly' neural networks, Natural Computing 1 (2002) 307–322.
[81] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.P. Schwefel, R. Manner (Eds.), Parallel Problem Solving from Nature –PPSN III, Lecture Notes in Computer Science, vol. 866, Springer, Berlin/ Heidelberg, 1994, pp. 249–257.
[82] K. Price, Eliminating drift bias from the differential evolution algorithm, in: U. Chakraborty (Ed.), Advances in Differential Evolution, Studies in Computational Intelligence, vol. 143, Springer, Berlin/Heidelberg, 2008, pp. 33–88.
[83] K. Price, R.M. Storn, J.A. Lampinen. Differential evolution: A practical approach to global optimization (Natural Computing Series). Secaucus, USA: Springer-Verlag New York, NJ, Inc., 2005.
[84] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[85] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[86] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, IEEE Congress on Evolutionary Computation 2005 (CEC 2005), vol. 2, 2005, pp. 1785–1791.
[87] J. Robinson, S. Sinton, Y. Rahmat-Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna, in: Antennas and Propagation Society International Symposium 2002, vol. 1, IEEE, 2002, pp. 314–317.
[88] A. Salman, A.P. Engelbrecht, M.G. Omran, Empirical analysis of self-adaptive differential evolution, European Journal of Operational Research 183 (2) (2007) 785–804.
[89] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, forth ed., Chapman & Hall/CRC, 2007.
[90] P. Spanevello, M.A. Montes de Oca, Experiments on adaptive heterogeneous pso algorithms, in: F. Hutter, M.A. Montes de Oca (Eds.), Proceedings of SLS-DS 2009: Doctoral Symposium on Engineering Stochastic Local Search Algorithms, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2009, pp. 36–40 (Technical Report No. TR/IRIDIA/2009-024).
[91] W.M. Spears, D.T. Green, D.F. Spears, Biases in particle swarm optimization, International Journal of Swarm Intelligence Research 1 (2) (2010) 34–57.
[92] R. Storn, K. Price, Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[93] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report; Nanyang Technological University AND KanGAL Report #2005005, IIT Kanpur, India, 2005.
[94] A.M. Sutton, M. Lunacek, L.D. Whitley, Differential evolution and non-separability: using selective pressure to focus search, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, USA, 2007, pp. 1428–1435.
[95] H. Talbi, M. Batouche, Hybrid particle swarm with differential evolution for multimodal image registration, in: IEEE International Conference on Industrial Technology, 2004, IEEE ICIT '04, vol. 3, 2004, pp. 1567–1572.
[96] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, Technical Report; Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.
[97] D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Parallel differential evolution, in: IEEE Congress on Evolutionary Computation, 2004 (CEC 2004), vol. 2, 2004, pp. 2023–2029.
[98] D.K. Tasoulis, V.P. Plagianakos, M.N. Vrahatis, Clustering in evolutionary algorithms to efficiently compute simultaneously local and global minima, in: IEEE Congress on Evolutionary Computation, 2005 (CEC 2005), vol. 2, 2005, pp. 1847–1854.
[99] R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: Hybridization perspectives and experimental illustrations, Applied Mathematics and Computation 217 (12) (2011) 5208–5226.
[100] S. Theodoridis, K. Koutroumbas, Pattern Recognition, forth ed., Academic Press, 2008.
[101] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation, 2004 (CEC 2004), vol. 2, 2004, pp. 1980–1987.
[102] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences 181 (20) (2011) 4515–4538.
[103] M. Weber, F. Neri, V. Tirronen, A study on scale factor in distributed differential evolution, Information Sciences 181 (12) (2011) 2488–2511.
[104] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Computing – A Fusion of Foundations, Methodologies and Applications 14 (11) (2009) 1187–1207.
[105] D. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[106] B. Xin, J. Chen, J. Zhang, H. Fang, Z. Peng. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 99 (2011) 1–24.
[107] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178 (15) (2008) 2985–2999.
[108] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: R. Matouek P.O., editor. Proceedings of MENDEL 2003, 9th International Mendel Conference on Soft Computing. 2003. p. 41–46..
[109] Zaharie D. A multipopulation differential evolution algorithm for multimodal optimization. In: R. Matousek, P. O (Eds.), Proceedings of Mendel 2004, 10th International Conference on Soft Computing, 2004, pp. 17–22.
[110] C. Zhang, J. Ning, S. Lu, D. Ouyang, T. Ding, A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization, Operations Research Letters 37 (2) (2009) 117–122.
[111] J. Zhang, A.C. Sanderson, Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization, Springer, 2009.
[112] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.

[113] W. Zhang, X. Xie, DEPSO: hybrid particle swarm with differential evolution operator, in: IEEE International Conference on Systems, Man and Cybernetics, 2003 (SMC 2003), vol. 4, 2003, pp. 3816–3821.
[114] S. Zhao, P.N. Suganthan, S. Das, Self-adaptive differential evolution with multi-trajectory search for large-scale optimization, Soft Computing – A Fusion of Foundations, Methodologies and Applications 15 (11) (2010) 2175–2185.