Swarm and Evolutionary Computation xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Global-best brain storm optimization algorithm

Mohammed El-Abd

Electrical and Computer Engineering Department, American University of Kuwait, P.O. Box 3323, Safat 13034, Kuwait

ARTICLE INFO

Keywords:
Brain storm optimization
Global-best
Per-variable updates
Re-initialization
Fitness-based grouping
Unconstrained optimization

ABSTRACT

Brain storm optimization (BSO) is a population-based metaheuristic algorithm that was recently developed to mimic the brainstorming process in humans. It has been successfully applied to many real-world engineering applications involving non-linear continuous optimization. In this work, we propose improving the performance of BSO by introducing a global-best version combined with per-variable updates and fitness-based grouping. In addition, the proposed algorithm incorporates a re-initialization scheme that is triggered by the current state of the population. The introduced Global-best BSO (GBSO) is compared against other BSO variants on a wide range of benchmark functions. Comparisons are based on final solutions and convergence characteristics. In addition, GBSO is compared against global-best versions of other meta-heuristics on recent benchmark libraries. Results prove that the proposed GBSO outperform previous BSO variants on a wide range of classical functions and different problem sizes. Moreover, GBSO outperforms other global-best meta-heuristic algorithms on the well-known CEC05 and CEC14 benchmarks.

1. Introduction

One class of algorithms used to solve non-linear continuous and/or discrete optimization problems is Population-based algorithms. Population-based algorithms maintain a population of individuals (solutions) and update them over a number of iterations (generations) until some stopping criterion is met. Population-based algorithms could be further categorized based on the inspiration behind their population update mechanism. The first category is evolutionary algorithms, in which the update process is inspired by the biological evolution process. These algorithms include Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary Strategies (ES), and Evolutionary Programming (EP). The second category includes swarm intelligence algorithms, in which the update process is inspired by some behavior of some living organism. A number of swarm intelligence algorithms are referred to as foraging algorithms as they mimic the foraging behavior of animals and/or insects. Examples of foraging algorithms include Particle Swarm Optimization (PSO) [1,2], Ant Colony Optimization [3], Artificial Bee Colony [4], and many more. Other swarm intelligence algorithms are inspired by different kinds of behaviors including for example the egg laying behavior of cuckoos in Cuckoo Search (CS) [5] and the echolocation behavior of bats in the Bat Algorithm (BA) [6].

The brain storm optimization (BSO) algorithm [7,8] is a population-based algorithm proposed to mimic brainstorming sessions held by humans. A typical brainstorming session involves gathering a group of experts having different backgrounds, expertise, and abilities in order to develop a solution for a problem at hand. Following such a process helps in successfully solving the tackled problem. The first version of the developed BSO algorithm had a number of disadvantages including the need to provide the number of clusters before hand, the computational complexity of the clustering stage, the lack of a re-initialization step, and the fixed schedule for updating the step size. Some of these disadvantages have already been addressed in the literature by either improving the clustering step [9–11], provide a better update method [9,12,13], or introduce a re-initialization mechanism [14–16]. However, to the best of our knowledge, a global-best version of BSO has not been proposed before.

In this paper we propose multiple modifications to improve the performance of BSO. These modifications include adopting a fitness-based grouping mechanism, using the global-best idea information for updating the population, and applying the update scheme on every problem variable separately. The proposed Global-best BSO (GBSO) is compared against three recent variants of BSO using a suite of 20 well-known benchmark functions. Moreover, GBSO is compared against the 2011 version of Standard PSO (SPSO) [17], Global-best guided ABC (GABC) [18] and the Improved Global-best Harmony Search (IGHS) [19] on the CEC05 [20] and the CEC14 [21] benchmarks for increased problem sizes.

The rest of the paper is organized as follows: Section 2 gives details about the BSO algorithm. Different improvements proposed in the literature to improve BSO are covered in Section 3. The proposed

E-mail address: melabd@auk.edu.kw.

http://dx.doi.org/10.1016/j.swevo.2017.05.001
Received 4 August 2016; Received in revised form 17 February 2017; Accepted 5 May 2017

GBSO is fully detailed in Section 4. Section 5 presents the experimental study and the reached results. Finally, the paper is concluded in Section 6.

2. Brain storm optimization

In BSO, a population is defined as a collection of *ideas*. A single *idea* represents a solution to the problem. In each iteration, a population of ideas (solutions) is updated. Initially, ideas are randomly scattered in the search space. In a single iteration, each idea $idea^i$ is updated as follows:

- First, k-means clustering is used to group similar ideas and the best idea in each cluster is saved as the *cluster center*,
- Second, BSO generates a new idea $nidea^i$ by setting it equal to one of the following:
  - A probabilistically selected cluster center,
  - A randomly selected idea from a probabilistically selected cluster,
  - The random combination of two probabilistically selected cluster centers, or
  - The random combination of two randomly selected ideas from two probabilistically selected clusters.

  One of these four operations is randomly selected based on a number of parameters $p_{one-cluster}$, $p_{one-center}$, and $p_{two-centers}$. Moreover, a cluster is probabilistically selected according to its size (i.e. the number of ideas in the cluster),
- Third, the generated $nidea^i$ is perturbed using a step-size parameter ξ and Gaussian distribution,
- Finally, $nidea^i$ replaces the current $idea^i$ if it has a better fitness. Otherwise, it will be discarded.

The BSO algorithm is shown in Fig. 1.

Algorithm 1. The BSO algorithm.

Require *MaxIterations*, n, m, $p_{one-cluster}$,$p_{one-center}$, and $p_{two-centers}$
Randomly initialize n ideas
Evaluate the n ideas
$iter$=1
while $iter \leq MaxIterations$ **do**
  Cluster n ideas into m clusters using k-means
  Rank ideas in each cluster and select cluster centers
  foreach idea i **do**
  if $rand < P_{one-cluster}$ **then**
  Probabilistically select a cluster c_r
  if $rand < P_{one-center}$ **then**
  $nidea^i = center_{c_r}$
  else
  Randomly select an idea j in cluster c_r
  $nidea^i = idea^j_{c_r}$
  end if
  else
  Probabilistically select two clusters c_{r1} and c_{r2}
  Randomly select two ideas c^j_{r1} and c^k_{r2}
  r=$rand$
  if $rand < P_{two-centers}$ **then**
  $nidea^i = r \times center_{c_{r1}} + (1 - r) \times center_{c_{r2}}$
  else
  $nidea^i = r \times idea^j_{c_{r1}} + (1 - r) \times idea^k_{c_{r2}}$
  end if
  end if
  $\xi = rand \times logsig(\frac{0.5 \times Max.Iterations - Current.Iteration}{k})$
  $nidea^i = nidea^i + \xi \times N(0, 1)$
  iffitness($nidea^i$) > fitness($idea^i$) **then**
  $idea^i$=$nidea^i$
  end if
  end for
  end while
  returnbest idea

Note that n is the population size, m is the number of clusters, $N(0, 1)$ represents a Gaussian distribution with mean 0 and a standard deviation of 1. *rand* is a uniformly distributed random number between 0 and 1. Finally, ξ is a dynamically updated step-size and k is for changing the slope of the *logsig* function. For more on BSO, interested readers can refer to [22].

3. Previous BSO improvements

A number of improvements have been proposed in the literature to improve the performance of BSO by addressing some of its disadvantages.

3.1. The clustering process

To overcome the burden of the clustering process, the authors in [9] used a Simple Grouping Strategy (SGM) instead of k-means clustering. In their approach, m seeds are selected randomly at each iteration, and then each one of the n ideas in the current population are assigned to the group of the nearest seed. In addition, perturbing the newly generated idea was done using an idea difference approach. Their method replaced the ξ parameter with a different factor p_r that controls injecting the open minded element, represented by randomly generated problem variables, into the idea creation process. Although their grouping strategy reduced the computational burden of k-means, it still requires a lot of distance calculations in the search space in order to assign different ideas to the different groups. Moreover, their grouping strategy has slightly sacrificed the performance on multi-modal functions. The proposed algorithm provided better results over PSO, DE, and BSO on a small set of classical functions.

Another attempt to overcome the burden of the clustering process was reported in [10]. The work introduced the idea of random grouping to minimize the clustering overhead. This is done by dividing the population into randomly constructed m clusters choosing the fittest idea in each group as its center. Although their grouping strategy aimed at mimicking random discussions between human individuals, it does not provide any ground basis or similarity measures for clustering. In other words, the random grouping strategy could be regarded as the exact opposite of k-means while the SGM previously explained lies somewhere in the middle.

To overcome the challenge of successfully presetting an appropriate number of clusters, the authors in [23] proposed to dynamically set the number of clusters using Affinity Propagation (AP) clustering. AP continuously changes the number of clusters according to their structure information. Although AP clustering still requires some computational effort as similarities need to be calculated between every two data points, the authors did not comment on the computational cost of their algorithm in comparison to using k-means. In addition, no information was given about how the *preference* for AP clustering was set although it has a direct effect on the generated number of clusters. Authors provided experiments showing how their algorithm provides good deployment of a Wireless Sensor Network.

Yet another approach to improve the clustering stage in BSO was recently proposed in [11]. The authors used Agglomerative Hierarchical Clustering (AHC) as it does not require pre-specifying the number of clusters. Moreover, the probability of creating new ideas using a single or multiple (two or three) clusters is adapted according to the quality of solutions generated. However, the authors did not provide details on how these individuals are generated. The developed

Table 1
Classical Functions – I.

Function	Definition	Range
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[−100,100]
Griewank	$f_2(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\frac{x_i}{\sqrt{i}} + 1$	[−600, 600]
Ackley	$f_3(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) + \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)$	[−32.768, 32.768]
Rastrigin	$f_4(x) = \sum_{i=1}^{D}[x_i - 10\cos(2\pi x_i) + 10]$	[−5.12,5.12]
Rosenbrock	$f_5(x) = \sum_{i=1}^{\frac{D}{2}}(100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2)$	[−2.048,2.048]
Schwefel 2.20	$f_6(x) = -\sum_{i=1}^{D}\lvert x_i\rvert$	[−100, 100]
Schwefel 2.21	$f_7(x) = \max_{1<i<D}\lvert x_i\rvert$	[−100, 100]
Schwefel 2.22	$f_8(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	[−100, 100]
Schwefel 2.23	$f_9(x) = \sum_{i=1}^{D} x_i^{10}$	[−10, 10]
Schwefel 2.25	$f_{10}(x) = \sum_{i=2}^{D}((x_i - 1)^2 + (x_1 - x_i^2)^2)$	[0,10]
Schwefel 2.26	$f_{11}(x) = 418.893 \times D - \frac{1}{D}\sum_{i=1}^{D} x_i\sin(\sqrt{\lvert x_i\rvert})$	[−500, 500]

Table 2
Classical Functions – II.

Function	Definition	Range
Powel-Sum	$f_{12}(x) = \sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	[−1, 1]
Levy	$f_{13}(x) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_i + 1)] + (w_D - 1)^2[1 + \sin^2(2\pi w_D)]$	[−10, 10]
Alpine 1	$f_{14}(x) = \sum_{i=1}^{D}\lvert x_i\sin(x_i) + 0.1x_i\rvert$	[−10, 10]
Alpine 2	$f_{15}(x) = \prod_{i=1}^{D}\sqrt{x_i}\sin(x_i)$	[0,10]
Pathological	$f_{16}(x) = \sum_{i=1}^{D-1}(0.5 + \frac{\sin^2(\sqrt{100x_i^2 + x_{i+1}^2}) - 0.5}{1 + 0.001(x_i^2 - 2x_ix_{i+1} + x_{i+1}^2)^2})$	[−100, 100]
Schaffer	$f_{17}(x) = \sum_{i=1}^{D} 0.5 + \frac{\sin^2(\sqrt{x_i^2 + x_{i+1}^2}) - 0.5}{[1 + 0.001(x_i^2 + x_{i+1}^2)]^2}$	[−100, 100]
Step 2	$f_{18}(x) = \sum_{i=1}^{D}(\lfloor x_i + 0.5\rfloor)^2$	[−100, 100]
Sum Squares	$f_{19}(x) = \sum_{i=1}^{D} ix_i^2$	[−10, 10]
Stretched V-Sine	$f_{20}(x) = \sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{0.25}[\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 0.1]$	[−10, 10]

Table 3
Individual Components Effect on Classical Functions – Results of 10D.

Bench.	BSO		FBG		Re-Init		Gbest		Per-Variable	
Func.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	9.69e−19	3.86e−18	6.58e−19	2.06e−18	9.79e−19	1.99e−18	**8.42e−21**	**1.51e−20**	4.49e−07	9.64e−07
2	1.74e+01	7.53e+00	1.88e+01	6.35e+00	**9.94e−01**	**4.20e−01**	1.39e+01	6.12e+00	2.57e+00	2.71e+00
3	1.00e−09	1.77e−09	4.12e−10	3.09e−10	6.76e−10	9.70e−10	**6.32e−11**	**8.87e−11**	3.92e−04	6.11e−04
4	5.87e+00	2.21e+00	4.97e+00	1.97e+00	8.52e+00	3.67e+00	6.47e+00	3.02e+00	**7.30e−01**	**6.88e−01**
5	3.71e+00	6.34e−01	**3.35e+00**	**5.74e−01**	**3.26e+00**	**7.40e−01**	4.17e+00	4.70e−01	6.56e+00	4.13e−01
6	1.91e−08	2.92e−08	3.19e−09	3.62e−09	2.08e−08	2.93e−08	**1.26e−09**	**3.48e−09**	3.18e−03	5.45e−03
7	2.57e−08	4.68e−08	**3.85e−09**	**3.78e−09**	3.46e−08	3.49e−08	**4.51e−09**	**7.76e−09**	1.35e−03	5.58e−03
8	1.09e+02	8.41e+01	2.47e+02	8.74e+01	1.55e+01	4.79e+01	1.20e+02	9.30e+01	**6.69e−03**	**4.55e−03**
9	1.95e−77	7.46e−77	7.39e−83	2.62e−82	4.22e−73	2.31e−72	**6.68e−86**	**3.26e−85**	9.06e−35	4.57e−34
10	5.43e−18	1.53e−17	1.23e−18	2.39e−18	1.08e−18	2.00e−18	**8.39e−21**	**3.37e−20**	4.81e−09	1.98e−08
11	1.71e+03	4.27e+02	1.71e+03	3.91e+02	1.50e+03	3.68e+02	1.65e+03	3.29e+02	**5.55e+02**	**2.60e+02**
12	1.99e−10	2.62e−10	1.12e−09	1.13e−09	4.22e−09	3.96e−09	2.53e−10	4.05e−10	**2.03e−11**	**4.04e−11**
13	1.32e+00	1.63e+00	3.50e+00	2.85e+00	1.88e−01	7.15e−01	8.02e−01	1.41e+00	**2.09e−07**	**8.38e−07**
14	5.88e−03	7.04e−03	3.46e−03	3.92e−03	1.88e−03	3.01e−03	3.74e−03	3.55e−03	**4.03e−04**	**3.57e−04**
15	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
16	1.22e+00	3.57e−01	1.89e+00	3.66e−01	9.76e−01	4.19e−01	1.28e+00	4.23e−01	**6.94e−01**	**3.46e−01**
17	2.55e+00	4.16e−01	3.53e+00	4.53e−01	2.09e+00	4.38e−01	2.33e+00	4.66e−01	**9.40e−01**	**4.32e−01**
18	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
19	1.82e−14	4.93e−14	1.79e−15	6.28e−15	1.50e−13	5.06e−13	**8.28e−17**	**3.60e−16**	5.18e−06	2.29e−05
20	1.14e+00	3.52e−01	1.20e+00	4.76e−01	7.88e−01	3.03e−01	1.05e+00	3.44e−01	**3.78e−01**	**1.83e−01**

Table 4
Individual Components Effect on Classical Functions – Results of 30D.

Bench.	BSO		FBG		Re-Init		Gbest		Per-Variable	
Func.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	5.49e−18	1.01e−17	1.42e−18	1.81e−18	4.94e−18	5.10e−18	**1.58e−20**	**2.45e−20**	1.52e+02	4.81e+02
2	1.19e−02	1.19e−02	**1.30e−02**	**1.25e−02**	**1.40e−02**	**1.70e−02**	**1.08e−02**	**9.93e−03**	2.75e+01	1.73e+01
3	1.27e−09	9.59e−10	6.88e−10	3.66e−10	1.53e−09	6.77e−10	**7.12e−11**	**3.88e−11**	6.73e−01	7.50e−01
4	4.07e+01	1.17e+01	3.66e+01	8.46e+00	5.00e+01	1.38e+01	4.46e+01	1.03e+01	**1.43e+01**	**4.73e+00**
5	2.61e+01	6.16e−01	2.55e+01	1.28e+00	**2.40e+01**	**2.23e+00**	2.55e+01	8.21e−01	2.74e+01	1.47e+00
6	1.19e−05	2.69e−05	**1.02e−07**	**1.61e−07**	4.97e−06	7.14e−06	**1.77e−07**	**4.42e−07**	7.32e+01	4.83e+01
7	1.46e−05	1.60e−05	**5.44e−07**	**4.87e−07**	1.30e−05	9.73e−06	2.30e−06	1.40e−06	8.22e+00	7.51e+00
8	5.43e+02	1.24e+02	7.17e+02	1.91e+02	**1.16e+02**	**1.74e+02**	5.46e+02	1.19e+02	1.30e+02	9.80e+01
9	3.86e−64	2.01e−63	**1.51e−74**	**4.10e−74**	1.44e−64	5.58e−64	8.13e−74	4.33e−73	3.28e−05	9.70e−05
10	1.31e−17	1.48e−17	3.44e−18	1.03e−17	8.10e−18	1.28e−17	**9.15e−21**	**1.98e−20**	3.36e−02	6.94e−02
11	5.81e+03	8.97e+02	5.40e+03	7.10e+02	5.63e+03	8.71e+02	5.73e+03	8.88e+02	**3.22e+03**	**4.40e+02**
12	3.41e−10	2.76e−10	1.66e−09	9.54e−10	6.37e−09	3.57e−09	6.28e−10	7.27e−10	**2.37e−10**	**3.60e−10**
13	1.70e+01	8.28e+00	1.94e+01	9.47e+00	5.59e+00	4.36e+00	1.40e+01	6.76e+00	**1.58e+00**	**1.77e+00**
14	1.49e−01	1.03e−01	5.99e−02	4.00e−02	4.27e−02	3.26e−02	9.59e−02	6.39e−02	5.42e−02	5.14e−02
15	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
16	5.54e+00	7.68e−01	7.05e+00	9.03e−01	4.78e+00	2.63e+00	5.24e+00	9.00e−01	**4.25e+00**	**7.01e−01**
17	1.08e+01	9.91e−01	1.17e+01	9.13e−01	8.98e+00	1.43e+00	1.02e+01	9.88e−01	**6.47e+00**	**1.05e+00**
18	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.24e+02	2.97e+02
19	4.49e−08	6.07e−08	2.15e−08	2.88e−08	2.08e−08	3.34e−08	**7.77e−09**	**7.87e−09**	9.17e−01	2.80e+00
20	4.54e+00	6.73e−01	4.79e+00	9.14e−01	4.37e+00	6.18e−01	4.50e+00	6.88e−01	**3.57e+00**	**1.04e+00**

Table 5
Individual Components Effect on CEC05 Functions – Results of 10D.

Bench.	BSO		FBG		Re-Init		Gbest		Per-Variable	
Func.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	2.65e−14	2.88e−14	**0.00e+00**	**0.00e+00**	2.84e−14	2.89e−14	3.79e−15	1.44e−14	6.53e−06	2.93e−05
2	2.31e−11	3.27e−11	**5.29e−12**	**1.02e−11**	1.15e−10	2.38e−10	**7.24e−12**	**1.02e−11**	1.82e+02	4.47e+02
3	4.78e+04	3.66e+04	4.98e+04	4.19e+04	**3.05e+04**	**2.64e+04**	5.40e+04	3.65e+04	7.04e+05	7.08e+05
4	6.67e+03	5.12e+03	7.04e+03	6.29e+03	**8.87e+00**	**3.88e+01**	5.23e+03	4.05e+03	2.05e+03	1.36e+03
5	**8.54e−07**	**1.29e−06**	**5.81e−07**	**1.91e−06**	1.93e−06	4.59e−06	**6.43e−07**	**1.34e−06**	2.01e+03	1.57e+03
6	1.02e+01	2.84e+01	8.59e+01	1.99e+02	**3.23e+00**	**3.44e+00**	6.45e+00	8.10e+00	2.28e+02	7.07e+02
7	1.30e+03	2.02e+01	4.46e+02	1.50e+02	**1.85e+02**	**8.08e+01**	**1.65e+02**	**7.62e+01**	1.52e+03	1.56e+02
9	6.40e+00	2.82e+00	5.44e+00	2.05e+00	9.49e+00	3.70e+00	7.00e+00	2.97e+00	**8.30e−01**	**7.43e−01**
10	7.23e+00	3.06e+00	**5.67e+00**	**1.59e+00**	1.03e+01	3.45e+00	7.86e+00	2.98e+00	1.90e+01	8.87e+00
11	2.01e+00	1.05e+00	2.45e+00	1.32e+00	**9.55e−01**	**8.85e−01**	2.55e+00	1.14e+00	8.79e+00	1.03e+00
12	**1.72e+02**	**3.84e+02**	**1.86e+02**	**4.62e+02**	**1.06e+02**	**3.27e+02**	4.82e+02	6.70e+02	2.91e+02	5.09e+02
13	6.22e−01	1.82e−01	7.21e−01	2.07e−01	1.03e+00	3.09e−01	6.64e−01	2.21e−01	**4.82e−01**	**1.31e−01**
14	3.94e+00	3.15e−01	4.20e+00	2.68e−01	3.46e+00	2.70e−01	3.91e+00	2.49e−01	**3.08e+00**	**4.10e−01**

Table 6
Individual Components Effect on CEC05 Functions – Results of 30D.

Bench.	BSO		FBG		Re-Init		Gbest		Per-Variable	
Func.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	1.44e−13	3.25e−14	**6.44e−14**	**1.97e−14**	1.21e−13	3.87e−14	8.72e−14	2.88e−14	9.37e+02	1.72e+03
2	5.48e−06	5.81e−06	1.81e−06	1.53e−06	2.48e−06	1.97e−06	**1.17e−06**	**1.07e−06**	1.94e+04	7.80e+03
3	4.84e+05	1.54e+05	4.88e+05	2.00e+05	**2.74e+05**	**1.33e+05**	5.45e+05	1.97e+05	4.06e+07	2.37e+07
4	2.29e+04	9.01e+03	3.34e+04	1.36e+04	**1.48e+03**	**1.68e+03**	1.67e+04	6.48e+03	2.67e+04	1.03e+04
5	2.99e+03	6.95e+02	2.21e+03	5.89e+02	**1.17e+03**	**3.86e+02**	3.02e+03	7.52e+02	1.12e+04	2.25e+03
6	5.09e+02	3.44e+02	5.16e+02	1.27e+03	**3.96e+02**	**3.88e+02**	**4.10e+02**	**3.96e+02**	5.00e+06	1.67e+07
7	4.80e+03	3.02e+01	4.42e+02	7.71e+01	1.09e+02	3.12e+01	**1.65e−01**	**5.77e−01**	6.83e+03	5.68e+02
9	5.09e+01	1.33e+01	3.50e+01	9.92e+00	6.32e+01	2.15e+01	5.20e+01	1.34e+01	**1.47e+01**	**6.19e+00**
10	3.79e+01	1.07e+01	**3.15e+01**	**7.99e+00**	6.64e+01	1.91e+01	4.62e+01	9.17e+00	1.83e+02	1.67e+01
11	1.19e+01	2.62e+00	**8.67e+00**	**2.90e+00**	**7.76e+00**	**2.52e+00**	1.24e+01	1.91e+00	3.96e+01	1.09e+00
12	3.80e+03	4.62e+03	4.21e+03	4.24e+03	**1.67e+03**	**2.30e+03**	4.09e+03	3.19e+03	6.46e+03	7.61e+03
13	3.30e+00	9.23e−01	3.27e+00	6.31e−01	3.95e+00	1.09e+00	3.15e+00	8.06e−01	**2.91e+00**	**8.76e−01**
14	1.34e+01	3.98e−01	1.37e+01	3.42e−01	**1.26e+01**	**4.13e−01**	1.31e+01	3.59e−01	**1.26e+01**	**3.77e−01**

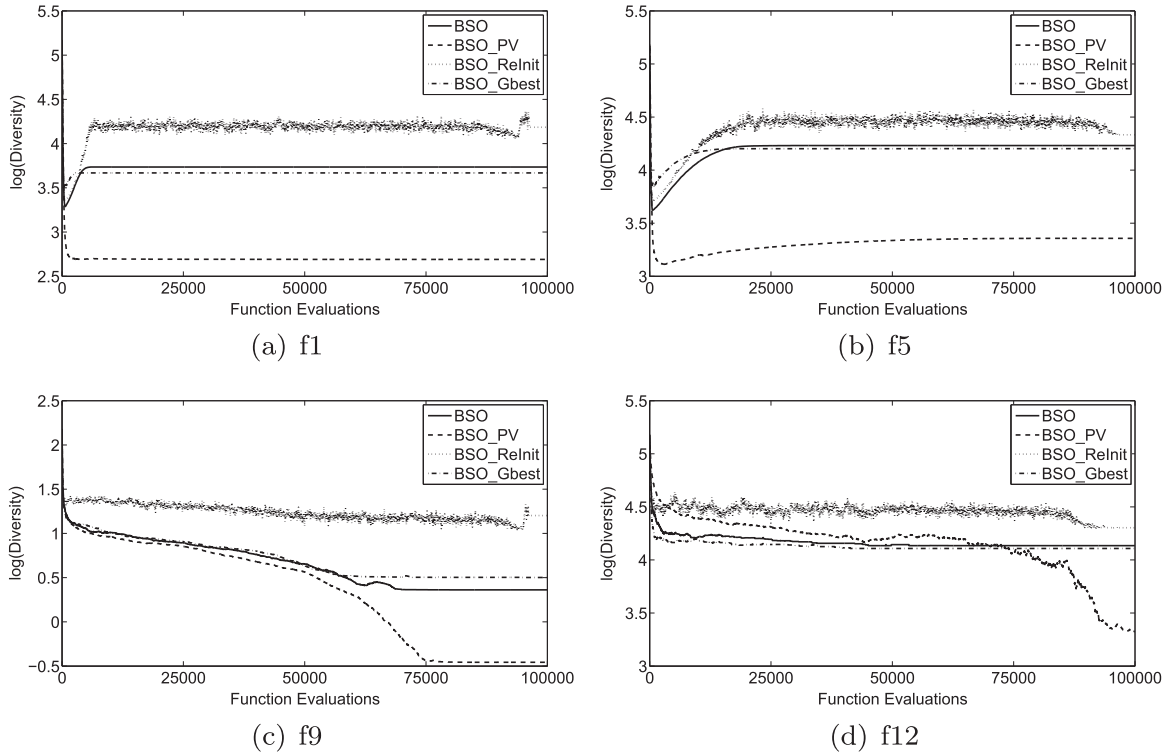

Fig. 1. Population diversity for individual components for a sample of the cec05 benchmark functions; 10 dimensions.

Table 7
Sensitivity to C_{min} and C_{max} on Classical Functions – Results of 10D.

Benchmark	**[0.1,0.9]**		**[0.2,0.8]**		**[0.3,0.7]**	
Function	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
1	**1.94e−21**	**3.29e−21**	8.42e−21	1.51e−20	9.25e−21	1.75e−20
2	1.27e+01	6.43e+00	1.39e+01	6.12e+00	1.14e+01	6.26e+00
3	**3.33e−11**	**9.90e−11**	6.32e−11	8.87e−11	5.89e−11	7.47e−11
4	6.80e+00	2.69e+00	6.47e+00	3.02e+00	6.47e+00	2.76e+00
5	4.28e+00	5.49e−01	4.17e+00	4.70e−01	4.06e+00	6.98e−01
6	**4.62e−10**	**1.02e−09**	1.26e−09	3.48e−09	1.99e−09	4.83e−09
7	**4.09e−09**	**6.46e−09**	**4.51e−09**	**7.76e−09**	8.32e−09	1.20e−08
8	**7.01e+01**	**8.88e+01**	1.20e+02	9.30e+01	**6.23e+01**	**8.41e+01**
9	1.11e−87	6.04e−87	6.68e−86	3.26e−85	**2.33e−88**	**1.22e−87**
10	**2.33e−21**	**8.09e−21**	8.39e−21	3.37e−20	7.97e−21	1.82e−20
11	1.50e+03	3.21e+02	1.65e+03	3.29e+02	1.72e+03	3.34e+02
12	3.46e−10	5.68e−10	2.53e−10	4.05e−10	3.37e−10	4.05e−10
13	1.36e+00	1.71e+00	8.02e−01	1.41e+00	7.18e−01	1.77e+00
14	6.56e−03	1.14e−02	3.74e−03	3.55e−03	5.28e−03	4.92e−03
15	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
16	1.35e+00	3.29e−01	1.28e+00	4.23e−01	1.19e+00	2.90e−01
17	2.27e+00	4.47e−01	2.33e+00	4.66e−01	2.31e+00	4.35e−01
18	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
19	**1.98e−17**	**5.39e−17**	8.28e−17	3.60e−16	4.27e−17	1.23e−16
20	1.13e+00	3.01e−01	1.05e+00	3.44e−01	1.13e+00	3.70e−01

algorithm improved the results over BSO on a number of classical functions.

3.2. Fixed step size

In addition to introducing the concept of random grouping, the authors in [10] have also modified the ξ update schedule as follows:

$$\xi = rand \times e^{1-\frac{Max.Iterations}{Max.Iterations-Current.Iteration+1}} \quad (1)$$

This modification was proposed as the authors indicated that the change of ξ in the old formula only takes effect for a very short interval. However, the new formula does not take the actual search space size into account.

A different approach to better adapt the fixed schedule for updating ξ, was proposed in [12] to update it using the dynamic range of ideas in each dimension. The authors used to different parameters ξ_i^{center} and $\xi_i^{individual}$ as the first one was used when generating an individual based on cluster center(s) while the second one was used when a new individual is generated based on a randomly selected individual. The authors introduced new parameters k_1 and k_2 for controlling the step size but provided no guidelines on how to set these parameters. The authors provided experiments illustrating how the step sizes ξ_i^{center} and $\xi_i^{individual}$ oscillate during the search in comparison to the rapid

Table 8
Sensitivity to C_{min} and C_{max} on CEC05 Functions – Results of 10D.

Benchmark	[0.1,0.9]		[0.2,0.8]		[0.3,0.7]	
Function	Mean	Std.	Mean	Std.	Mean	Std.
1	1.89e−15	1.04e−14	3.79e−15	1.44e−14	5.68e−15	1.73e−14
2	**7.59e−12**	**1.16e−11**	**7.24e−12**	**1.02e−11**	1.74e−11	4.16e−11
3	4.56e+04	3.77e+04	5.40e+04	3.65e+04	4.92e+04	3.37e+04
4	5.81e+03	3.28e+03	5.23e+03	4.05e+03	5.87e+03	3.82e+03
5	**4.40e−07**	**1.05e−06**	**6.43e−07**	**1.34e−06**	1.29e−06	3.27e−06
6	1.88e+01	5.98e+01	6.45e+00	8.10e+00	4.28e+00	2.16e+00
7	1.70e+02	6.21e+01	1.65e+02	7.62e+01	1.67e+02	7.25e+01
9	8.06e+00	3.47e+00	7.00e+00	2.97e+00	6.63e+00	2.68e+00
10	**6.83e+00**	**2.29e+00**	7.86e+00	2.98e+00	**6.10e+00**	**2.29e+00**
11	2.21e+00	1.05e+00	2.55e+00	1.14e+00	2.08e+00	1.12e+00
12	4.38e+02	5.95e+02	4.82e+02	6.70e+02	2.85e+02	5.75e+02
13	7.87e−01	2.01e−01	6.64e−01	2.21e−01	7.63e−01	2.05e−01
14	3.85e+00	2.64e−01	3.91e+00	2.49e−01	3.86e+00	2.97e−01

Table 9
Different Gbest Application Techniques on CEC05 Functions – Results of 10D.

Benchmark	0.75		1.5		[0.2,0.8]	
Function	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00e+00**	**0.00e+00**	7.58e−15	1.97e−14	3.79e−15	1.44e−14
2	4.20e−12	5.72e−12	**1.63e−12**	**3.11e−12**	7.24e−12	1.02e−11
3	**3.86e+04**	**2.32e+04**	8.04e+04	6.60e+04	5.40e+04	3.65e+04
4	5.00e+03	2.91e+03	**4.71e−02**	**2.58e−01**	5.23e+03	4.05e+03
5	**1.27e−07**	**1.95e−07**	8.13e−07	3.89e−06	6.43e−07	1.34e−06
6	1.37e+02	6.51e+02	1.55e+01	4.87e+01	**6.45e+00**	**8.10e+00**
7	8.65e+01	3.45e+01	**1.82e+01**	**1.21e+01**	1.65e+02	7.62e+01
9	8.60e+00	4.08e+00	1.06e+01	5.49e+00	**7.00e+00**	**2.97e+00**
10	1.03e+01	4.90e+00	1.35e+01	7.36e+00	**7.86e+00**	**2.98e+00**
11	**2.46e+00**	**1.40e+00**	3.50e+00	1.66e+00	**2.55e+00**	**1.14e+00**
12	7.34e+02	1.19e+03	8.91e+02	2.80e+03	**4.82e+02**	**6.70e+02**
13	8.59e−01	3.49e−01	9.69e−01	3.23e−01	**6.64e−01**	**2.21e−01**
14	3.79e+00	3.56e−01	**3.52e+00**	**2.85e−01**	3.91e+00	2.49e−01

Table 10
Different Gbest Application Techniques on CEC05 Functions – Results of 30D.

Benchmark	0.75		1.5		[0.2,0.8]	
Function	Mean	Std.	Mean	Std.	Mean	Std.
1	**5.68e−14**	**2.57e−29**	8.53e−14	3.25e−14	8.72e−14	2.88e−14
2	2.39e−06	1.80e−06	**1.03e−06**	**1.16e−06**	**1.17e−06**	**1.07e−06**
3	4.78e+05	1.58e+05	5.43e+05	2.25e+05	5.45e+05	1.97e+05
4	2.73e+04	1.02e+04	**1.47e+04**	**6.01e+03**	**1.67e+04**	**6.48e+03**
5	3.40e+03	6.00e+02	3.87e+03	1.04e+03	**3.02e+03**	**7.52e+02**
6	4.11e+02	3.69e+02	4.10e+02	4.39e+02	4.10e+02	3.96e+02
7	4.93e+01	2.33e+01	**2.39e−02**	**1.73e−02**	1.65e−01	5.77e−01
9	**6.17e+01**	**2.11e+01**	8.07e+01	2.85e+01	**5.20e+01**	**1.34e+01**
10	**5.00e+01**	**1.83e+01**	8.74e+01	2.84e+01	**4.62e+01**	**9.17e+00**
11	**1.18e+01**	**3.09e+00**	1.50e+01	4.47e+00	**1.24e+01**	**1.91e+00**
12	3.35e+03	4.14e+03	4.14e+03	4.12e+03	4.09e+03	3.19e+03
13	**3.20e+00**	**9.28e−01**	5.03e+00	1.63e+00	**3.15e+00**	**8.06e−01**
14	1.33e+01	4.10e−01	1.31e+01	4.29e−01	1.31e+01	3.59e−01

decrease of the original step size.

The work in [16] modified Eq. (1) by multiplying it with a parameter α that is proportional to width of the search domain.

$$\xi = rand \times e^{1-\frac{Max.Iterations}{Max.Iterations-Current.Iteration+1}} \times \alpha \quad (2)$$

3.3. The update process

In addition to dynamically updating the step size, the work in [12] modified the update process as well. In their approach, $3n$ new ideas were created in each iteration. n ideas were created based on n randomly selected cluster centers using the step size ξ_i^{center}, n ideas were generated using the combination of $2n$ randomly selected cluster centers using the step size ξ_i^{center}, and n ideas were generated using the combination of $2n$ randomly selected individuals using the step size $\xi_i^{individual}$. The update process started by randomly grouping a total of $4n$ ideas (the old n ideas and the new $3n$ ideas) into n groups of 4 ideas each. Finally, the best idea in each group was copied to the next generation. Evaluating new ideas and replacing old ideas was implemented in batch mode. The developed algorithm improved the results over BSO on a number of classical functions.

To improve the update process, the authors in [13] introduced a

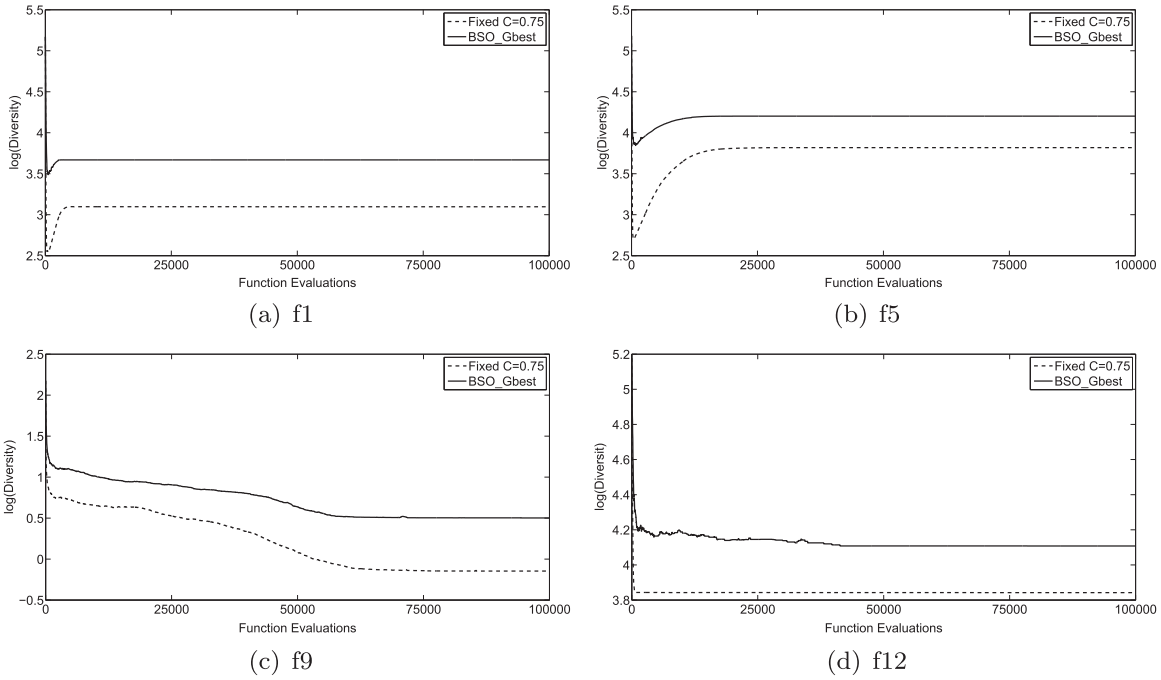

Fig. 2. Population diversity for individual components for a sample of the cec05 benchmark functions; 10 dimensions.

Table 11
Performance of BSO Variants on Classical Functions – Results of 10D.

Benchmark	BSODE		RGBSO		IRGBSO		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	5.49e−18	1.40e−17	**2.06e−18**	**5.22e−18**	7.20e−18	1.01e−17	8.35e−17	3.90e−17
2	1.05e−01	4.42e−02	2.09e+01	7.17e+00	**4.17e−02**	**2.98e−02**	6.21e−02	3.48e−02
3	2.64e−09	2.53e−09	**8.25e−10**	**8.58e−10**	1.13e−09	1.46e−09	3.24e−09	8.56e−10
4	6.81e+00	3.09e+00	4.54e+00	1.91e+00	5.33e+00	2.93e+00	**0.00e+00**	**0.00e+00**
5	3.50e+00	8.88e−01	3.29e+00	7.96e−01	**1.48e−01**	**5.19e−02**	7.83e−01	2.44e−01
6	8.86e−08	1.99e−07	**4.74e−09**	**5.42e−09**	1.13e−08	1.33e−08	1.81e−08	4.38e−09
7	1.61e−07	3.23e−07	**6.35e−09**	**6.17e−09**	1.76e−08	2.08e−08	**5.82e−09**	**1.85e−09**
8	1.10e+01	2.55e+01	2.28e+02	1.41e+02	**1.10e−08**	**1.31e−08**	**1.31e−08**	**2.84e−09**
9	1.04e−72	5.20e−72	1.44e−82	4.74e−82	1.28e−87	5.98e−87	**2.45e−92**	**3.78e−92**
10	9.89e−18	1.91e−17	**1.82e−18**	**3.35e−18**	1.09e−16	1.31e−16	**2.91e−18**	**1.20e−18**
11	1.54e+03	3.74e+02	1.86e+03	3.48e+02	1.05e+03	2.53e+02	**1.27e−04**	**0.00e+00**
12	1.83e−09	2.67e−09	9.19e−10	1.20e−09	4.72e−10	4.20e−10	**3.93e−12**	**9.06e−12**
13	1.26e−12	4.20e−12	3.16e+00	2.87e+00	1.45e+00	1.92e+00	**6.68e−19**	**7.43e−19**
14	6.78e−05	2.49e−04	4.88e−03	4.43e−03	6.57e−04	1.50e−03	**2.08e−10**	**3.52e−11**
15	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
16	1.21e+00	3.34e−01	1.81e+00	4.65e−01	9.52e−01	5.17e−01	**3.85e−01**	**2.01e−01**
17	1.94e+00	6.50e−01	3.51e+00	3.39e−01	1.41e+00	5.50e−01	**1.38e−01**	**6.26e−02**
18	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
19	1.13e−13	2.91e−13	2.95e−15	6.46e−15	3.52e−16	5.33e−16	**5.09e−18**	**3.12e−18**
20	1.03e+00	2.00e−01	1.28e+00	4.50e−01	**3.68e−01**	**3.33e−01**	**2.24e−01**	**1.21e−01**

new idea generation mechanism borrowed from differential evolution while also using the step-size schedule proposed in [10]. In their work, if the newly created idea was to follow a randomly selected idea from a randomly selected cluster cr, this operator (referred to as intra-cluster operator) was modified as follows (where F is the mutation scaling factor while r_1 and r_2 are mutually exclusive integers randomly chosen in the selected cluster):

$$nidea^i = center_{cr} + F \times (idea_{cr}^{r1} - idea_{cr}^{r2}) \quad (3)$$

And if the generated idea was to follow a combination of two randomly selected ideas j and k from two randomly selected clusters $cr1$ and $cr2$, this operator (referred to as inter-cluster operator) was modified as follows (where F is defined as above while $GlobalIdea$ is the best idea in the population):

$$nidea^i = GlobalIdea + F \times (idea_{cr1}^{j} - idea_{cr2}^{k}) \quad (4)$$

Their approach improved the performance over BSO while having a faster speed of convergence. It also had comparable performance with PSO, CoDE [24], and SaDE [25] on the CEC05 benchmarks [20].

3.4. Lack of a re-initialization behavior

The authors in [14,15] proposed enhancing the population diversity of BSO by re-initializing a small part of the population every a predetermined number of iterations. The authors conducted a number of experiments to study the effect of changing the percentage of ideas being re-initialized by re-initializing a fixed percentage of the population, an increasing percentage, and a decreasing percentage. In general, the performance of BSO was improved and the population diversity

Table 12
Performance of BSO Variants on Classical Functions – Results of 30D.

Benchmark	BSODE		RGBSO		IRGBSO		GBSO	
Function	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
1	3.05e−17	6.57e−17	**2.23e−18**	**2.90e−18**	2.32e−17	2.59e−17	1.57e−16	4.03e−17
2	9.16e−03	8.87e−03	1.24e−02	1.30e−02	**5.02e−03**	**8.07e−03**	**4.83e−03**	**7.34e−03**
3	2.43e−09	1.74e−09	**9.91e−10**	**5.39e−10**	**8.55e−10**	**3.58e−10**	2.93e−09	3.65e−10
4	3.69e+01	7.89e+00	3.70e+01	8.86e+00	3.96e+01	1.12e+01	**4.98e−04**	**2.33e−03**
5	2.52e+01	1.03e+00	2.54e+01	5.90e−01	**2.00e+01**	**9.08e−01**	2.21e+01	1.08e+00
6	1.72e−05	2.48e−05	2.34e−07	2.95e−07	5.72e−07	8.92e−07	**4.94e−08**	**5.81e−09**
7	9.74e−05	1.14e−04	7.00e−07	5.95e−07	1.47e−06	9.69e−07	**7.43e−08**	**6.88e−08**
8	2.38e+02	1.81e+02	7.62e+02	1.64e+02	3.62e−07	4.51e−07	**5.11e−08**	**7.16e−09**
9	2.04e−61	4.16e−61	8.72e−70	4.31e−69	2.24e−73	1.12e−72	**3.41e−91**	**4.29e−91**
10	3.25e−17	4.69e−17	**8.29e−18**	**1.12e−17**	1.28e−15	1.22e−15	**6.81e−18**	**1.52e−18**
11	5.39e+03	5.86e+02	5.35e+03	7.24e+02	3.12e+03	3.38e+02	**3.82e−04**	**2.21e−19**
12	3.81e−09	3.45e−09	8.54e−10	6.66e−10	9.12e−10	4.09e−10	**1.32e−13**	**2.02e−13**
13	1.82e−02	9.09e−02	1.91e+01	7.05e+00	1.00e+01	5.84e+00	**3.82e−18**	**4.38e−18**
14	1.71e−02	1.29e−02	6.20e−02	3.38e−02	5.79e−03	7.40e−03	**7.18e−10**	**8.62e−10**
15	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
16	6.47e+00	1.23e+00	6.99e+00	1.04e+00	6.05e+00	1.94e+00	**2.29e+00**	**5.24e−01**
17	8.78e+00	8.53e−01	1.20e+01	1.07e+00	8.49e+00	8.09e−01	**1.23e+00**	**3.93e−01**
18	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
19	6.89e−07	1.05e−06	2.90e−08	5.39e−08	9.21e−10	1.69e−09	**6.30e−12**	**1.82e−11**
20	4.43e+00	8.64e−01	4.70e+00	9.92e−01	**1.53e+00**	**4.33e−01**	**1.34e+00**	**4.15e−01**

Table 13
Performance of BSO Variants on Classical Functions – Results of 50D.

Benchmark	BSODE		RGBSO		IRGBSO		GBSO	
Function	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
1	**3.18e−17**	**4.02e−17**	**4.92e−17**	**5.97e−17**	**2.94e−17**	**2.52e−17**	1.58e−16	2.68e−17
2	1.97e−03	3.63e−03	1.18e−03	3.27e−03	**5.92e−04**	**2.05e−03**	4.83e−03	6.69e−03
3	3.68e−09	2.79e−09	3.20e−09	1.79e−09	**1.17e−09**	**5.17e−10**	2.32e−09	2.55e−10
4	6.64e+01	1.12e+01	6.34e+01	1.62e+01	8.36e+01	1.76e+01	**4.09e−01**	**5.44e−01**
5	4.56e+01	1.30e+00	4.62e+01	1.36e+00	**4.07e+01**	**9.59e−01**	4.25e+01	6.34e−01
6	1.39e−03	3.89e−03	4.53e−03	1.13e−02	7.26e−06	1.05e−05	**7.44e−08**	**7.84e−09**
7	2.17e−03	1.25e−03	2.81e−03	1.89e−03	**1.63e−05**	**1.08e−05**	3.64e−05	3.60e−05
8	5.92e+02	2.88e+02	6.41e+02	2.27e+02	9.98e−06	1.39e−05	**9.12e−08**	**1.14e−08**
9	1.06e−54	3.36e−54	2.78e−54	1.18e−53	3.56e−71	1.69e−70	**2.20e−89**	**7.92e−89**
10	9.29e−17	9.02e−17	9.00e−17	8.98e−17	3.15e−15	2.62e−15	**5.67e−18**	**1.83e−18**
11	9.45e+03	1.07e+03	9.43e+03	9.10e+02	4.76e+03	5.72e+02	**9.11e−01**	**4.35e+00**
12	2.85e−09	2.23e−09	2.30e−09	1.79e−09	6.84e−10	2.86e−10	**3.34e−14**	**5.54e−14**
13	7.63e−02	1.69e−01	7.24e−02	1.59e−01	1.29e+01	6.67e+00	**1.01e−17**	**2.28e−17**
14	8.92e−02	6.53e−02	9.55e−02	6.73e−02	1.20e−02	1.47e−02	**4.13e−05**	**2.06e−04**
15	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
16	1.30e+01	1.84e+00	1.36e+01	1.44e+00	1.15e+01	2.92e+00	**4.17e+00**	**7.37e−01**
17	1.56e+01	1.36e+00	1.58e+01	1.48e+00	1.65e+01	1.13e+00	**2.75e+00**	**7.11e−01**
18	8.00e−02	2.77e−01	2.40e−01	4.36e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
19	3.27e−04	5.26e−04	2.97e−04	4.20e−04	1.36e−06	1.71e−06	**3.12e−08**	**3.25e−08**
20	7.85e+00	1.17e+00	8.56e+00	1.16e+00	4.84e+00	1.41e+00	**3.16e+00**	**3.85e−01**

was increased by their proposed strategy. However, their re-initialization mechanism was not triggered by the current state of the population as it was applied at fixed intervals. At the same time, a fixed number of ideas were being re-initialized even if some them did not converge. The authors provided experiments showing how the diversity of their proposed algorithm is higher than the diversity of the population in the original BSO. In addition, their algorithm provided better results over BSO on a set of classical functions.

Another attempt to address the lack of a re-initialization process in BSO was reported in [16]. The work introduced the use of a re-initialization scheme borrowed from the Artificial Bee Colony (ABC) algorithm [26]. A re-initialization process is carried out when a certain idea has not been improved for a specified number of iterations, *Threshold*. In addition, the idea could be either re-initialized using uniform distribution or by using a differential approach as follows (where F is defined as above while r_1, r_2 and r_3 are mutually exclusive integers randomly chosen between 1 and n):

$$nidea^i = idea^{r1} + F \times (idea^{r2} - idea^{r3}) \quad (5)$$

The proposed algorithm provided better results over the works in [10,13] on the CEC15 benchmarks [27]. However, no experiments were provided on the performance of the individual re-initialization and step size update components.

4. Proposed Global-best BSO (GBSO)

In this section we explain the rationale behind several algorithmic design decisions taken to enhance the performance of BSO. Note that we use the same re-initialization stage and the step-size updating scheme used in [16].

- **Fitness-based grouping:** During the grouping stage, the ideas are ranked according to their fitness and grouping is done to ensure that good and bad ideas are equally distributed among the different

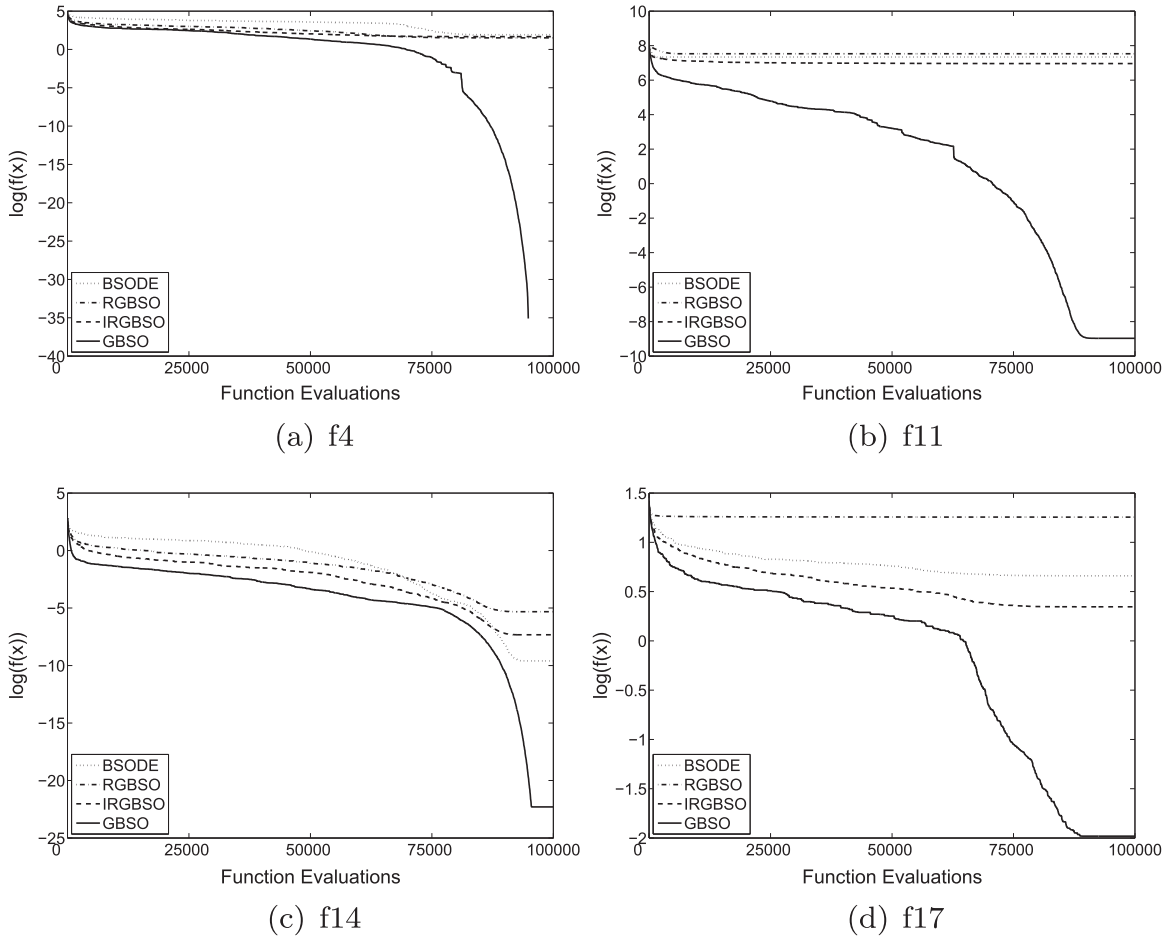

Fig. 3. Convergence behavior of all the algorithms for a sample of the classical benchmark functions; 10 dimensions.

groups. As stated earlier, *k*-means is at one end of the spectrum with maximum computational cost and distance-based clustering while random grouping is at the other end with minimum computational cost and no similarity measure at all. Fitness-based grouping will lie somewhere in the middle as it provides less computational complexity than *k*-means and a better distribution of ideas among the different groups when compared to random grouping. Fitness-based grouping would also provide less computational cost than AP or AHC. Moreover, identifying the center of each cluster becomes very simple as the center would be the first idea in the cluster. Fitness-based grouping is shown in Fig. 2 where % is the remainder operator.

Algorithm 2. Fitness-based grouping.

Require n, m
1: Rank ideas according to fitness in descending order
2: **for**each idea i **do**
3: $g = (i-1)\%m + 1$
4: Add idea i to group g
5: **end for**

- **Per-variable updates:** In the original BSO and all of its subsequent variants, new ideas are generated by updating all problem variables in one step. In this work, we propose generating new ideas one problem variable at a time. This means that the first problem variable could be updated using the center of one randomly selected cluster, while the next problem variable is updated using the combination of two randomly selected ideas from two randomly selected clusters, and so on. In the original BSO, only one idea contributes to the generation of a new idea, with a probability $p_{one-cluster}$, or at most two ideas, with a probability $1 - p_{one-cluster}$. While in the proposed approach, multiple ideas contribute in generating the new idea allowing for more cooperation among the population individuals.
- **The global-best update:** Here we borrow the global-best guidance concept originally proposed in PSO and subsequently applied in many algorithms. The incorporation of the global-best information in the update equation has improved the performance of many meta-heuristic algorithms. The influence of the best idea in the population is added after the new idea is generated as follows:

$$nidea^i = nidea^i + rand(1, DimSize) \times C \times (GlobalBest - nidea^i) \quad (6)$$

However, in the original BSO, the best m positions already contribute to the update process. This probability is equal to $p_{one-cluster} \times p_{one-center} + (1 - p_{one-cluster}) \times p_{two-centers}$. When adopting the parameter settings used in most BSO literature, this probability is equal to 0.42. That is why the influence of the global-best update equation in our version is gradually increased though the search process. This is done by updating C according to the following equation:

$$C = C_{min} + \frac{CurrentIteration}{MaxIterations} \times (C_{max} - C_{min}) \quad (7)$$

- **The re-initialization step:** The same re-initialization scheme previously proposed in [16] is still employed in this version of the algorithm. Not only it will help in increasing the population diversity and prevent premature convergence, it also serves the principle of

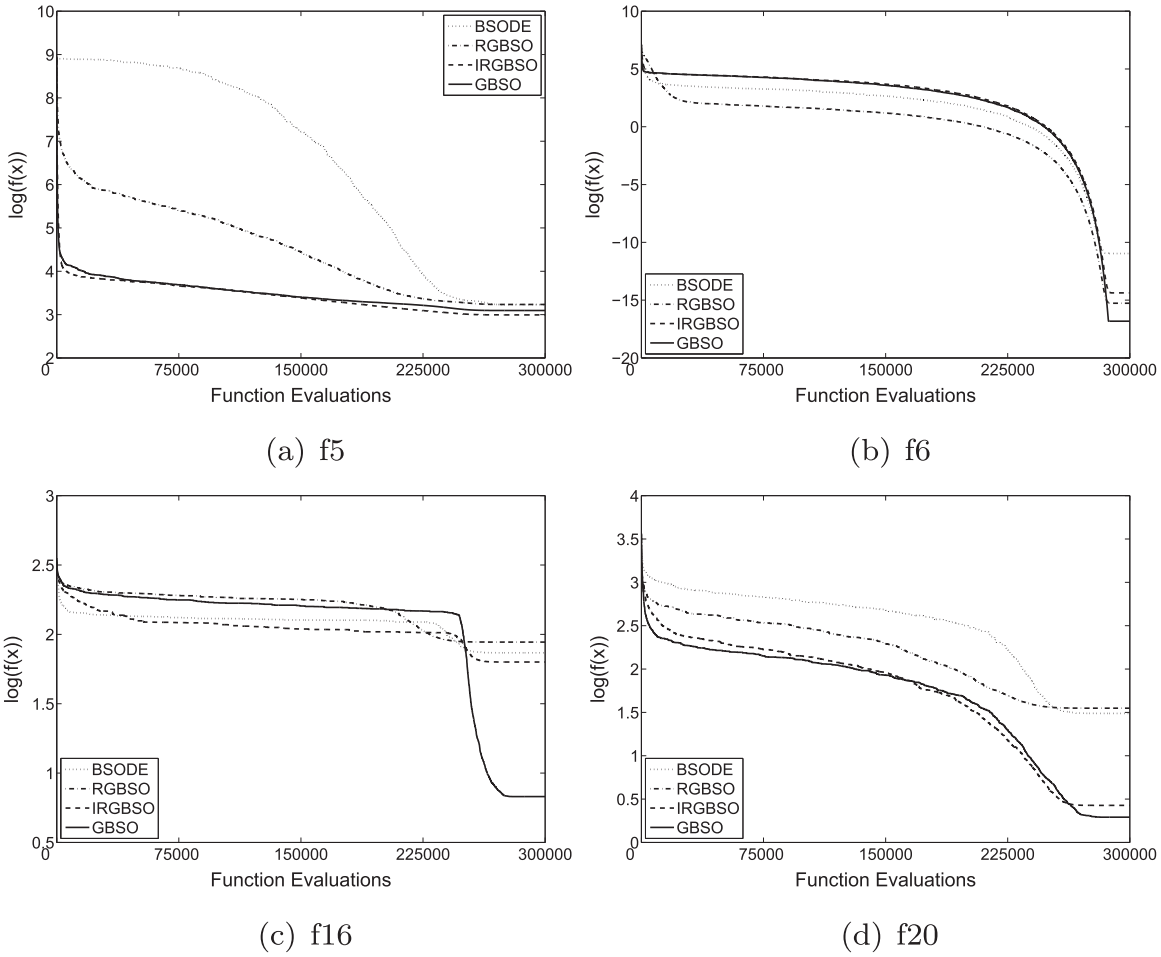

Fig. 4. Convergence behavior of all the algorithms for a sample of the classical benchmark functions; 30 dimensions.

injecting open minded elements as proposed in [9] but for the entire idea instead of randomly selected problem variables.

- **Population update:** The population is updated once at the end. The update mechanism is still the same where each newly generated idea $nidea^i$ replaces the old one $idea^i$ if it has a better fitness. The population update is carried after all new ideas are generated to minimize the computational complexity of the algorithm as calculating the objective function for the whole population in MATLAB would be faster than calculating the objective function for a single individual at a time.

The full GBSO algorithm is shown in Fig. 3.

Algorithm 3. The GBSO algorithm.

Require *MaxIterations*, n, m, $p_{one-cluster}$,$p_{one-center}$, and $p_{two-centers}$, *Step*, C_{min}, C_{max}

Randomly initialize n ideas
Evaluate the n ideas
iter=1
while $iter \leq MaxIterations$ **do**
  Perform fitness-based grouping
  foreach idea i **do**
  foreach problem variable j **do**
  if $rand < P_{one-cluster}$ **then**
  Probabilistically select a cluster c_r
  if $rand < P_{one-center}$ **then**
  $nidea^{ij} = center^j_{c_r}$
  else
  Randomly select an idea k in cluster c_r
  $nidea^{ij} = idea^{kj}_{c_r}$
  end if
  else
  Randomly select two clusters c_{r1} and c_{r2}
  Randomly select two ideas $c^{k_1}_{r1}$ and $c^{k_2}_{r2}$
  r=*rand*
  if $rand < P_{two-centers}$ **then**
  $nidea^{ij} = r \times center^j_{c_{r1}} + (1 - r) \times center^j_{c_{r2}}$
  else
  $nidea^{ij} = r \times idea^{k_1 j}_{c_{r1}} + (1 - r) \times idea^{k_2 j}_{c_{r2}}$
  end if
  end if
end for
  $C = C_{min} + \frac{Current.Iteration - 1}{Max.Iterations} \times (C_{max} - C_{min})$
  if $C < rand$ **then**
  $nidea^i = nidea^i + rand(1, DimSize) \times C \times (GlobalBest - nidea^i)$
  end if
  $\xi = rand \times e^{1 - \frac{Max.Iterations}{Max.Iterations - Current.Iteration + 1}} \times Step$
  $nidea^i = nidea^i + \xi \times N(\mu, 0)$
  end for
  Update population
end while
returnbest idea

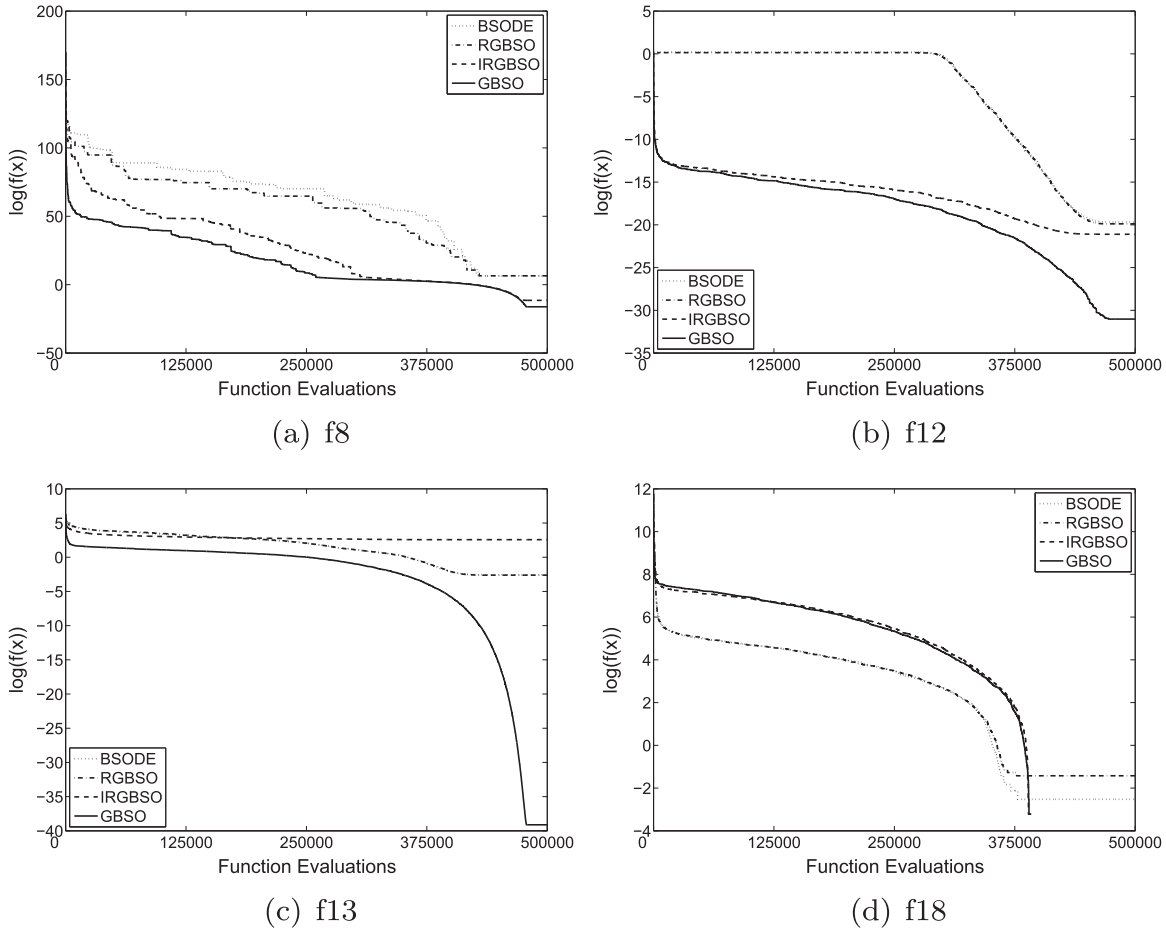

Fig. 5. Convergence behavior of all the algorithms for a sample of the classical benchmark functions; 50 dimensions.

Table 14
Statistical Tests Results For The BSO Variants.

Algorithm	Friedman Ranking	Holm *p*-value	Finner *p*-value
RGBSO	3.06	0	0
BSODE	3.03	0	0
IRGBSO	2.23	0.0196	0.0196
GBSO	**1.7**	–	–

5. Experimental results

5.1. Experimental procedure and parameters settings

To fully evaluate the performance of the proposed GBSO, the following set of experiments are conducted:

- In the first set of experiments, the effect of the different proposed components is identified using a set of 20 classical benchmark functions shown in Tables 1, 2 as well as the CEC05 benchmarks,
- In the second set of experiments, GBSO is compared against BSODE [13], RGBSO [10] and IRGBSO [16] using the classical functions,

Table 15
Performance of Global-best Algorithms on CEC05 Functions – Results of 10D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	8.18e−09	2.21e−09	**0.00e+00**	**0.00e+00**
2	6.71e+00	6.30e+00	**9.47e−15**	**2.15e−14**	1.13e−08	3.67e−09	9.39e−11	2.42e−10
3	6.23e+05	3.08e+05	**3.85e+04**	**2.91e+04**	7.54e+04	5.81e+04	4.97e+04	3.81e+04
4	6.64e+02	4.14e+02	**1.71e−14**	**2.65e−14**	1.34e−08	4.39e−09	2.97e−09	8.06e−09
5	2.03e+00	1.56e+00	**0.00e+00**	**0.00e+00**	2.75e−03	1.00e−03	1.42e−06	1.00e−06
6	**7.63e−02**	**9.26e−02**	4.11e+01	9.71e+01	3.19e+01	5.40e+01	6.21e+00	2.60e+00
7	**7.95e−02**	**5.09e−02**	9.46e−02	1.10e−01	1.64e−01	7.69e−02	**8.46e−02**	**5.13e−02**
9	**0.00e+00**	**0.00e+00**	5.24e+00	2.10e+00	1.63e−06	3.50e−07	**0.00e+00**	**0.00e+00**
10	1.35e+01	3.25e+00	**4.57e+00**	**2.09e+00**	9.68e+00	3.98e+00	**4.61e+00**	**2.34e+00**
11	5.23e+00	9.11e−01	3.27e+00	1.65e+00	9.92e−01	9.34e−01	**2.57e−01**	**5.35e−01**
12	**1.96e+02**	**1.05e+02**	2.00e+04	9.50e+03	**1.78e+02**	**4.63e+02**	5.44e+02	7.14e+02
13	**1.36e−01**	**1.18e−01**	7.93e−01	1.68e−01	4.39e−01	1.19e−01	4.84e−01	1.12e−01
14	3.21e+00	1.75e−01	2.30e+00	5.10e−01	3.28e+00	6.37e−01	**1.94e+00**	**5.35e−01**

Table 16
Performance of Global-best Algorithms on CEC05 Functions – Results of 30D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	6.82e−14	2.31e−14	**5.49e−14**	**1.04e−14**	8.87e−08	1.25e−08	**5.49e−14**	**1.04e−14**
2	1.38e+03	8.51e+02	**3.35e−13**	**1.07e−13**	6.51e−07	1.68e−07	3.15e−04	2.10e−04
3	6.22e+06	3.21e+06	**2.80e+05**	**1.37e+05**	4.78e+05	2.36e+05	1.06e+06	3.61e+05
4	3.29e+04	5.04e+03	5.03e+01	3.27e+01	**9.59e−03**	**1.89e−02**	1.83e−01	1.66e−01
5	7.62e+03	1.46e+03	4.73e+03	7.93e+02	1.18e+03	5.32e+02	**8.90e+01**	**7.94e+01**
6	**1.53e+01**	**2.67e+01**	5.04e+02	1.06e+03	1.61e+02	1.71e+02	8.80e+01	1.47e+02
7	2.77e−02	1.84e−02	2.94e−02	2.43e−02	**8.70e−03**	**1.06e−02**	**7.72e−03**	**7.91e−03**
9	**5.68e−14**	**2.57e−29**	5.77e+01	2.71e+01	1.69e−05	1.70e−06	3.60e−01	6.30e−01
10	1.70e+02	2.45e+01	5.52e+01	1.31e+01	4.98e+01	1.41e+01	**2.52e+01**	**7.85e+00**
11	2.69e+01	1.50e+00	2.66e+01	4.48e+00	5.60e+00	2.25e+00	**1.35e+00**	**1.74e+00**
12	6.54e+03	2.60e+03	1.04e+06	1.51e+05	**1.59e+03**	**1.74e+03**	4.44e+03	5.34e+03
13	**8.07e−01**	**1.56e−01**	5.98e+00	3.30e+00	1.23e+00	2.43e−01	1.73e+00	2.73e−01
14	1.27e+01	2.36e−01	1.20e+01	6.68e−01	1.19e+01	5.73e−01	**1.01e+01**	**8.59e−01**

Table 17
Performance of Global-best Algorithms on CEC05 Functions – Results of 50D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	1.65e−13	2.73e−14	**9.47e−14**	**2.73e−14**	2.74e−07	2.87e−08	**1.00e−13**	**2.45e−14**
2	1.39e+04	6.08e+03	**1.38e−11**	**1.64e−11**	9.63e−06	1.92e−06	1.07e−01	4.10e−02
3	1.18e+07	3.69e+06	**4.33e+05**	**1.44e+05**	9.07e+05	3.72e+05	2.09e+06	7.49e+05
4	9.28e+04	1.10e+04	3.92e+03	1.02e+03	3.18e+03	2.03e+03	**8.46e+01**	**4.50e+01**
5	2.13e+04	1.97e+03	1.24e+04	1.87e+03	3.29e+03	7.67e+02	**6.61e+02**	**3.23e+02**
6	**3.32e+01**	**4.35e+01**	6.77e+02	1.73e+03	1.45e+02	1.70e+02	1.44e+03	2.42e+03
7	1.02e−02	1.66e−02	1.12e−02	1.48e−02	**1.07e−03**	**2.80e−03**	1.48e−03	3.01e−03
9	**1.04e−13**	**2.15e−14**	1.31e+02	2.78e+01	5.12e−05	4.33e−06	2.85e+00	1.19e+00
10	5.25e+02	6.61e+01	1.52e+02	3.02e+01	8.94e+01	2.13e+01	**5.13e+01**	**1.24e+01**
11	5.33e+01	2.90e+00	5.72e+01	4.13e+00	1.20e+01	3.35e+00	**3.56e+00**	**2.43e+00**
12	2.96e+04	1.15e+04	5.16e+06	7.99e+05	**1.22e+04**	**1.04e+04**	1.79e+04	1.37e+04
13	**1.45e+00**	**2.45e−01**	1.38e+01	8.27e+00	2.01e+00	2.83e−01	3.19e+00	5.29e−01
14	2.24e+01	2.46e−01	**2.16e+01**	**6.81e−01**	**2.11e+01**	**7.82e−01**	**2.11e+01**	**9.18e−01**

Table 18
Statistical Tests Results For The Global-best Algorithms – CEC05 Functions.

Algorithm	Friedman Ranking	Holm p-value	Finner p-value
GABC	2.88	0.0064	0.0064
SPSO	2.81	0.0100	0.0075
IGHS	2.32	0.2542	0.2542
GBSO	**1.99**	–	–

- In the third and fourth set of experiments, GBSO is compared against the 2011 version of Standard PSO (SPSO) [17], Global-best guided ABC (GABC) [18] and the Improved Global-best Harmony Search (IGHS) [19] on the CEC05 and the CEC14 [21] benchmarks,
- Finally, GBSO is compared to some state-of-the-art algorithms on the CEC05 and CEC14 benchmarks.

For all BSO variants tested, we use the same parameters settings employed in the literature having the population size n=25, the number of clusters m=5, $p_{one-cluster} = 0.8$, $p_{one-center} = 0.4$, and $p_{two-centers} = 0.5$. For both GBSO and IRGBSO, we use the same parameter settings in [16] having *Threshold*=10, F=0.5, and $\alpha = 0.05 \times (UB - LB)$, where LB and UB are the lower and upper bounds of the search space. For GBSO, the values $C_{min} = 0.2$ and $C_{max} = 0.8$ are used. All experiments are conducted on dimensions D=10, D=30, and D=50 for a maximum of $10000 \times D$ function evaluations. In all tables, best results are highlighted in bold based on the non-parametric Wilcoxon-ranksum test with a 5% confidence interval.

Table 19
Number of Successful Runs For The Global-best Algorithms – CEC05 Functions.

Benchmark	Problem	GABC	SPSO	IGHS	GBSO
Function	Size				
1	10	30	30	30	30
2		0	30	30	30
4		0	30	30	30
5		0	30	0	10
6		8	0	0	0
7		0	2	0	0
9		30	0	30	30
10		0	0	0	1
11		0	0	0	19
12		0	0	3	0
13		6	0	0	0
1	30	30	30	30	30
2		0	30	28	0
6		2	0	0	0
7		9	12	21	20
9		30	0	30	19
1	50	30	30	30	30
2		0	30	0	0
7		21	19	30	30
9		30	0	30	1

5.2. Individual components effect

Results of the first set of experiments are provided in Tables 3 and 4 for the classical functions and Tables 5 and 6 for the CEC05 benchmarks.

Table 20
Performance Rates For The Global-best Algorithms – CEC05 Functions.

Benchmark	Problem	GABC	SPSO	IGHS	GBSO
Function	**Size**				
1	10	1.00	1.24	15.2	17.9
2		–	1.00	5.66	6.64
4		–	1.00	4.23	4.92
5		–	1.00	10.7	11.4
9		1.00	–	7.95	11.8
1	30	1.13	1.00	16.7	17.6
2		–	1.00	2.62	–
7		4.54	1.00	1.60	2.89
9		1.00	–	6.93	15.5
1	50	1.18	1.00	17.2	17.4
7		2.55	1.00	2.12	3.53
9		1.00	–	6.22	253.00

Inspecting the performance on the classical functions shows that the *per-variable update* and *Gbest* components provide the best performance on a large number of functions. On the other hand, the *re-initialization* component provides the best performance on the CEC05 benchmarks. This illustrates that there is no single component emerges as the obvious winner and that the performance of the different components varies on different benchmarks. Another important observation is that the adoption of each individual component by itself can outperform the original BSO algorithm.

When the second best local optima is far in the search space from the global optima, *FBG* could be very much similar to the distance-based *k*-means clustering as both of these optima will be classified in different clusters. That is why the performance when adopting this component is very similar to the performance of the original BSO on the Schwefel 2.26 (f11) classical function and its shifted version (f12) in the CEC05 benchmarks.

Fig. 1 shows how the diversity of the population evolves over the entire search process for the original BSO, the *per-variable update* component, the *Gbest* component, and the *Re – initilaization* component. The population diversity is measured according to the following formula:

$$Diveristy = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\sum_{j=1}^{D} x_{ij}^2 - \overline{x_j}^2} \tag{8}$$

where $\overline{x}$ is the mean position of the population.

The figure shows again that different components have different effects on the population. While the *Gbest* component does not have a big effect on diversity, the *re-initialization* component helps in increasing the diversity of the population by discarding stagnate ideas. Such an increase in diversity becomes essential for difficult benchmarks (involving shifting and rotation) as the CEC05. As for the *per-variable update* component, the level of cooperation induced by this component seems to have a negative effect on diversity for uni-modal functions. For functions f1 and f5, the diversity quickly decreases and remains very low till the end. On the other hand, for multi-modal functions f9 and f12, the diversity is maintained at a high level for a big part of the search process and gradually decreases towards the end.

Tables 7 and 8 present results of experiments carried to test the *Gbest* component sensitivity to the values of C_{min} and C_{max}. Rows with no bold entries mean that all results are statistically equivalent according to the non-parametric Wilcoxon-ranksum statistical test. Results show that the algorithm's performance is insensitive to the tested values of these parameters and that the choice of our setting to [0.2, 0.8] is suitable.

Moreover, Tables 9 and 10 present a comparison between our proposed *Gbest* mechanism and the use of a global-best Eq. (6), with different values for *C*, without having the increasing effect presented in

Table 21
Performance of Global-best Algorithms on CEC14 Functions – Results of 10D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
1	9.84e+04	7.41e+04	**2.04e+04**	**1.08e+04**	2.84e+04	1.63e+04	4.75e+04	4.85e+04
2	**6.91e+01**	**1.32e+02**	9.51e+02	1.06e+03	4.61e+03	3.88e+03	2.23e+03	2.15e+03
3	**1.05e+02**	**1.08e+02**	1.24e+03	7.08e+02	3.16e+02	6.85e+02	4.70e+02	7.70e+02
4	**3.13e–03**	**4.18e–03**	2.36e+01	1.61e+01	2.83e+01	1.33e+01	2.61e+01	1.49e+01
5	1.68e+01	7.35e+00	1.88e+01	5.11e+00	2.00e+01	9.79e–04	**1.57e+01**	**8.32e+00**
6	8.96e–01	4.99e–01	**3.18e–01**	**6.19e–01**	1.80e+00	1.43e+00	**4.50e–01**	**8.71e–01**
7	**4.81e–03**	**6.78e–03**	2.13e–02	1.76e–02	1.49e–01	6.88e–02	5.59e–02	2.93e–02
8	**0.00e+00**	**0.00e+00**	5.21e+00	2.32e+00	4.21e–09	1.25e–09	3.48e–02	1.61e–01
9	4.09e+00	1.37e+00	4.44e+00	2.10e+00	9.98e+00	4.69e+00	**2.95e+00**	**1.21e+00**
10	**3.31e–02**	**3.46e–02**	3.57e+02	1.81e+02	2.04e–01	8.82e–02	1.97e+00	2.40e+00
11	**1.58e+02**	**1.06e+02**	4.28e+02	2.76e+02	3.92e+02	1.98e+02	**1.04e+02**	**8.01e+01**
12	2.15e–01	6.48e–02	5.39e–01	1.59e–01	2.00e–02	3.33e–02	**8.37e–03**	**1.37e–02**
13	1.09e–01	1.83e–02	**6.16e–02**	**2.72e–02**	7.92e–02	2.53e–02	**5.05e–02**	**2.12e–02**
14	**9.71e–02**	**2.42e–02**	1.13e–01	5.30e–02	1.16e–01	5.35e–02	1.31e–01	6.04e–02
15	**6.22e–01**	**1.25e–01**	9.33e–01	1.97e–01	**7.80e–01**	**3.17e–01**	8.03e–01	2.54e–01
16	1.72e+00	3.48e–01	1.84e+00	4.35e–01	2.25e+00	5.91e–01	**1.20e+00**	**5.79e–01**
17	1.22e+05	1.02e+05	**1.51e+03**	**1.51e+03**	**2.53e+03**	**2.13e+03**	**1.39e+03**	**1.66e+03**
18	**5.88e+02**	**5.42e+02**	1.66e+03	2.63e+03	1.31e+04	1.09e+04	8.18e+03	6.48e+03
19	**1.52e–01**	**6.33e–02**	2.24e+00	5.71e–01	1.36e+00	7.27e–01	8.63e–01	3.31e–01
20	3.80e+02	5.84e+02	**1.84e+02**	**2.48e+02**	2.38e+03	2.76e+03	3.53e+02	8.12e+02
21	5.91e+03	5.89e+03	1.03e+03	1.06e+03	2.72e+03	3.00e+03	**2.93e+02**	**3.18e+02**
22	**1.87e–01**	**1.18e–01**	2.86e+01	7.29e+00	5.56e+01	7.22e+01	2.63e+01	4.49e+01
23	**2.70e+02**	**1.22e+02**	3.29e+02	0.00e+00	3.29e+02	0.00e+00	3.29e+02	2.30e–13
24	1.13e+02	2.71e+00	1.12e+02	4.32e+00	1.21e+02	1.68e+01	**1.07e+02**	**3.99e+00**
25	**1.25e+02**	**4.75e+00**	1.81e+02	2.74e+01	1.78e+02	3.41e+01	1.78e+02	3.17e+01
26	**9.73e+01**	**1.56e+01**	1.00e+02	2.56e–02	1.10e+02	3.05e+01	1.00e+02	1.97e–02
27	**9.85e+01**	**1.58e+02**	2.61e+02	1.49e+02	3.53e+02	8.17e+01	2.49e+02	1.14e+02
28	**3.62e+02**	**5.84e+00**	3.96e+02	5.39e+01	4.41e+02	1.08e+02	4.30e+02	6.51e+01
29	**2.97e+02**	**4.02e+01**	4.91e+02	1.39e+02	6.58e+05	8.83e+05	1.21e+05	4.11e+05
30	**5.27e+02**	**6.75e+01**	7.75e+02	3.02e+02	8.50e+02	2.88e+02	**5.76e+02**	**1.16e+02**

Table 22
Performance of Global-best Algorithms on CEC14 Functions – Results of 30D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	4.95e+06	3.93e+06	**2.74e+05**	**1.55e+05**	**2.70e+05**	**1.22e+05**	8.38e+05	4.94e+05
2	**4.70e+01**	**7.18e+01**	7.48e+03	5.10e+03	1.45e+04	1.20e+04	8.55e+03	7.48e+03
3	5.61e+02	5.04e+02	4.51e+03	1.54e+03	**8.27e–01**	**9.30e–01**	2.22e+02	2.96e+02
4	3.94e+01	3.15e+01	2.39e+01	3.27e+01	**6.50e+00**	**2.57e+01**	6.15e+01	1.83e+01
5	2.02e+01	5.89e–02	2.07e+01	9.98e–02	**2.00e+01**	**1.53e–03**	2.03e+01	3.33e–01
6	1.30e+01	1.79e+00	1.21e+01	2.50e+00	4.87e+00	2.21e+00	**9.84e–01**	**1.18e+00**
7	**1.81e–08**	**4.73e–08**	1.06e–02	1.28e–02	9.03e–03	8.64e–03	9.17e–03	9.68e–03
8	**1.14e–13**	**1.03e–28**	4.35e+01	1.04e+01	4.35e–08	5.59e–09	1.70e–01	4.16e–01
9	5.38e+01	9.20e+00	4.66e+01	1.28e+01	4.13e+01	1.02e+01	**2.31e+01**	**6.82e+00**
10	8.86e–01	1.57e+00	2.68e+03	5.63e+02	**2.31e–01**	**5.53e–02**	3.80e+00	2.27e+00
11	1.70e+03	2.68e+02	3.59e+03	6.74e+02	1.89e+03	5.12e+02	**4.37e+02**	**2.56e+02**
12	1.90e–01	5.71e–02	1.39e+00	2.93e–01	2.92e–02	1.70e–02	**1.86e–02**	**2.07e–02**
13	2.08e–01	2.62e–02	1.88e–01	4.04e–02	2.70e–01	5.24e–02	**1.57e–01**	**4.36e–02**
14	**1.82e–01**	**1.58e–02**	2.21e–01	4.08e–02	3.06e–01	2.20e–01	2.21e–01	4.03e–02
15	4.69e+00	9.99e–01	6.92e+00	2.69e+00	**2.94e+00**	**5.85e–01**	**3.38e+00**	**7.13e–01**
16	**9.16e+00**	**4.43e–01**	1.10e+01	4.66e–01	9.81e+00	8.51e–01	**8.66e+00**	**7.46e–01**
17	2.03e+06	1.34e+06	**2.12e+04**	**1.47e+04**	**2.50e+04**	**2.11e+04**	7.28e+04	4.65e+04
18	5.24e+03	5.75e+03	**1.44e+03**	**1.55e+03**	4.84e+03	7.14e+03	**2.39e+03**	**3.24e+03**
19	7.09e+00	9.24e–01	1.36e+01	2.53e+00	7.04e+00	1.96e+00	**4.72e+00**	**1.02e+00**
20	5.56e+03	2.13e+03	8.33e+02	4.31e+02	**7.52e+01**	**3.24e+01**	**8.80e+01**	**3.10e+01**
21	2.44e+05	1.75e+05	**2.01e+04**	**1.58e+04**	**1.28e+04**	**1.10e+04**	2.32e+04	1.53e+04
22	**2.39e+02**	**1.15e+02**	2.79e+02	9.59e+01	3.78e+02	1.69e+02	**2.24e+02**	**9.10e+01**
23	3.16e+02	6.50e–01	**3.15e+02**	**6.39e–05**	**3.15e+02**	**5.78e–14**	**3.15e+02**	**1.55e–04**
24	2.19e+02	1.74e+01	2.34e+02	7.04e+00	**2.12e+02**	**1.55e+01**	**2.00e+02**	**5.98e–02**
25	2.08e+02	1.36e+00	2.14e+02	2.47e+00	2.05e+02	1.47e+00	**2.03e+02**	**3.05e–01**
26	**1.00e+02**	**5.86e–02**	1.27e+02	4.49e+01	1.51e+02	6.59e+01	**1.00e+02**	**3.90e–02**
27	4.08e+02	2.88e+00	6.09e+02	1.24e+02	4.99e+02	9.23e+01	**3.96e+02**	**7.11e+01**
28	8.40e+02	3.39e+01	1.23e+03	2.82e+02	8.99e+02	1.09e+02	**7.94e+02**	**7.80e+01**
29	**1.18e+03**	**2.25e+02**	3.08e+06	6.24e+06	2.01e+06	3.70e+06	1.71e+05	1.21e+06
30	3.02e+03	8.46e+02	5.84e+03	1.79e+03	2.48e+03	9.09e+02	**2.03e+03**	**6.51e+02**

Table 23
Performance of Global-best Algorithms on CEC14 Functions – Results of 50D.

Benchmark	GABC		SPSO		IGHS		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	1.10e+07	3.89e+06	**7.92e+05**	**2.51e+05**	**9.04e+05**	**3.73e+05**	2.22e+06	7.45e+05
2	**6.74e+03**	**8.87e+03**	**7.77e+03**	**7.89e+03**	**7.75e+03**	**9.27e+03**	**9.26e+03**	**8.21e+03**
3	6.72e+03	2.82e+03	1.17e+03	3.34e+02	**3.61e–02**	**5.41e–02**	1.59e+03	8.57e+02
4	**6.34e+01**	**2.98e+01**	7.76e+01	3.35e+01	9.35e+01	8.77e+00	9.66e+01	2.90e+00
5	2.01e+01	7.40e–02	2.09e+01	9.24e–02	**2.00e+01**	**8.46e–03**	2.09e+01	3.73e–01
6	2.74e+01	2.10e+00	3.06e+01	3.13e+00	1.05e+01	3.54e+00	**2.08e+00**	**1.65e+00**
7	**2.56e–04**	**8.99e–04**	6.55e–03	1.31e–02	4.77e–03	5.33e–03	3.04e–03	4.34e–03
8	**2.20e–13**	**2.88e–14**	1.14e+02	2.23e+01	1.16e–07	1.64e–08	1.47e+00	1.30e+00
9	1.23e+02	1.43e+01	1.28e+02	2.53e+01	9.26e+01	1.92e+01	**4.76e+01**	**1.06e+01**
10	9.62e–01	1.25e+00	5.92e+03	9.42e+02	**2.24e–01**	**4.37e–02**	6.76e+00	6.01e+00
11	3.98e+03	4.50e+02	6.95e+03	1.09e+03	3.37e+03	6.70e+02	**1.58e+03**	**5.25e+02**
12	1.84e–01	3.90e–02	2.15e+00	3.93e–01	2.64e–02	1.40e–02	**1.80e–02**	**1.30e–02**
13	**2.65e–01**	**3.86e–02**	3.48e–01	7.08e–02	3.88e–01	5.82e–02	**2.56e–01**	**4.56e–02**
14	**2.19e–01**	**1.40e–02**	2.87e–01	3.33e–02	4.46e–01	3.24e–01	2.36e–01	6.71e–02
15	1.22e+01	1.79e+00	2.56e+01	7.36e+00	**6.55e+00**	**1.46e+00**	**6.36e+00**	**1.25e+00**
16	**1.73e+01**	**5.65e–01**	2.04e+01	5.29e–01	**1.75e+01**	**1.06e+00**	**1.71e+01**	**8.47e–01**
17	3.11e+06	1.59e+06	**3.72e+04**	**2.02e+04**	5.99e+04	2.89e+04	1.89e+05	1.51e+05
18	2.19e+03	1.45e+03	**1.86e+03**	**1.06e+03**	2.58e+03	1.70e+03	**1.51e+03**	**1.45e+03**
19	1.84e+01	2.56e+00	5.51e+01	1.94e+01	1.67e+01	2.55e+00	**1.15e+01**	**1.37e+00**
20	2.71e+04	6.94e+03	8.25e+02	2.55e+02	**1.80e+02**	**7.05e+01**	**1.84e+02**	**4.61e+01**
21	1.84e+06	8.64e+05	**4.64e+04**	**2.22e+04**	**5.32e+04**	**3.62e+04**	1.44e+05	7.87e+04
22	6.80e+02	1.98e+02	7.57e+02	2.50e+02	6.11e+02	1.96e+02	**2.74e+02**	**1.40e+02**
23	3.46e+02	2.28e+00	**3.44e+02**	**3.59e–02**	**3.44e+02**	**6.35e–05**	**3.44e+02**	**1.60e–03**
24	2.59e+02	2.49e+00	2.88e+02	6.81e+00	2.62e+02	5.42e+00	**2.56e+02**	**2.51e+00**
25	2.15e+02	1.64e+00	2.31e+02	4.46e+00	2.11e+02	2.80e+00	**2.07e+02**	**8.45e–01**
26	**1.00e+02**	**7.85e–02**	1.63e+02	4.89e+01	1.47e+02	5.06e+01	**1.32e+02**	**4.68e+01**
27	1.07e+03	1.83e+02	1.15e+03	1.22e+02	6.93e+02	1.27e+02	**4.35e+02**	**5.76e+01**
28	1.35e+03	1.33e+02	2.87e+03	7.01e+02	1.38e+03	2.85e+02	**1.07e+03**	**7.14e+01**
29	**2.00e+03**	**6.43e+02**	4.79e+07	7.40e+07	2.39e+06	9.07e+06	2.46e+03	6.22e+02
30	**9.92e+03**	**7.28e+02**	2.71e+04	5.73e+03	**1.04e+04**	**1.76e+03**	**9.11e+03**	**6.40e+02**

Table 24
Statistical Tests Results For The Global-best Algorithms – CEC14 Functions.

Algorithm	Friedman Ranking	Holm p-value	Finner p-value
SPSO	3.05	0	0
IGHS	2.64	0.0016	0.0012
GABC	2.32	0.0941	0.0941
GBSO	**1.99**	–	–

Eq. (7). Results show that when D=10, having the proposed increased influence of the global-best factor provides better results on multi-modal functions. For D=30, the results of having the increased *Gbest* effect improved on uni-modal functions while having the same competitive performance on multi-modal functions.

As for the effect on diversity, Fig. 2 illustrates the diversity of the population when adopting the increased Gbest mechanism versus the fixed application of the Gbest equation with C=0.75. The figure shows that our proposed approach maintains a higher level of diversity through the entire search.

5.3. GBSO vs. previous BSO variants

For the second set of experiments, results of all algorithms are provided in Tables 11–13. Results illustrate the superior performance of GBSO over previous BSO variants as it provides an improvement of several orders of magnitude on multiple benchmark functions.

Convergence curves of all algorithms for a sample of the functions across all problem sizes are presented in Figs. 3, 4 and 5. The curves show that GBSO has a desirable property as it continues to improve during the entire search process for a number of functions in different dimensions.

Based on suggestions provided in [28], further assessment of the performance of GBSO is performed using the Friedman test, to provide the ranking, and the post hoc procedures (Holm and Finner), to compare all algorithms against the best one. The tests are run using the software available at [29] and results are shown in Table 14 where the best (lowest) rank is highlighted in bold. For the Holm and Finner procedures, adjusted p-values are reported with significant differences at a 0.05 level of significance. The results confirm the superior performance of the proposed GBSO algorithm when compared to BSODE, RGBSO and IRGBSO.

5.4. GBSO vs. other global-best algorithms

For the third set of experiments, results of all algorithms are provided in Tables 15–17. Results for GBSO are provided in italic if it's the second best performer. Average rankings and p-values provided by the post hoc procedures (Holm and Finner) are given in Table 18. Note that function f8 was excluded from the comparisons as it has the global optimum near the bounds and it acts as the needle-in-hay-stack problem. All algorithms applied to this function practically reach the same solution.

Results show that for all problem sizes, SPSO is the best performer for uni-modal functions and is closely followed by GBSO. However, GBSO and GABC are the best performers on multi-modal functions. The 2007 version of SPSO was also proven to be very powerful on uni-modal functions while the original ABC was proven to be the best on multi-modal functions compared to many other algorithms in [30]. This illustrates that GBSO can match the global-best versions of powerful algorithms on functions with varying characteristics. Results of the post hoc experiments show that GBSO is the best performer overall for the CEC05 benchmarks.

Finally, comparing the results of GBSO to the results of its individual components in Tables 5 and 6 above shows that GBSO provides better results than any of its components on 8 functions out of 13 for D=10 and 10 functions out of 13 in D=30.

The number of successful runs for all algorithms is given in Table 19. A successful run is defined as a run in which the algorithm has reached the threshold of 10^{-6} for uni-modal functions and 10^{-2} for multi-modal functions as was set in [20]. For D=10, SPSO and GBSO are the most successful in uni-modal functions. For multi-modal functions, GBSO has a competitive performance being the only algorithm able to solve functions f10 and f11. For D=30 and D=50, SPSO is still the best performer on uni-modal functions. As for multi-modal functions, GBSO and IGHS are the most successful on function f7 while GABC and IGHS are the most successful on function f9.

Performance rates for all algorithms are provided in Table 20. The performance rate is defined as:

$$PerformanceRate = \frac{FES_{avg} \times TotalRuns}{SuccessfulRuns} \tag{9}$$

where FES_{avg} is the average number of function evaluations needed to reach the predetermined threshold taken over the successful runs only.

Results show that SPSO is usually the algorithm with the highest speed of convergence. SPSO is usually followed by either GABC or IGHS. In most functions, GBSO has the slowest speed of convergence. The slow speed of convergence of GBSO is due to its re-initialization scheme (not in SPSO or IGHS) in addition to the gradual use of the Gbest information (not in SPSO or GABC).

For the fourth set of experiments, results of all algorithms are provided in Tables 21–23. Average rankings and p-values provided by

Table 25
Comparison with CMA-ES on CEC05 Functions – Results of 10D and 50D.

Benchmark	D=10				D=50			
	CMA-ES		GBSO		CMA-ES		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	5.20e–09	1.994e–09	**0.00e+00**	**0.00e+00**	5.87e–09	8.59e–10	**1.00e–13**	**2.45e–14**
2	4.70e–09	1.56e–09	**9.39e–11**	**2.42e–10**	**7.86e–09**	**7.24e–10**	1.07e–01	4.10e–02
3	**5.60e–09**	**1.93e–09**	4.97e+04	3.81e+04	**6.14e–09**	**6.86e–10**	2.09e+06	7.49e+05
4	5.02e–09	1.71e–09	**2.97e–09**	**8.06e–09**	4.68e+05	3.11e+05	**8.46e+01**	**4.50e+01**
5	**6.58e–09**	**2.17e–09**	1.42e–06	1.00e–06	**2.85e+00**	**4.32e+00**	6.61e+02	3.23e+02
6	**4.87e–09**	**1.66e–09**	6.21e+00	2.60e+00	**7.13e–09**	**1.11e–09**	1.44e+03	2.42e+03
7	**3.31e–09**	**2.02e–09**	5.46e–02	5.13e–02	**7.22e–09**	**1.03e–09**	1.48e–03	3.01e–03
9	2.39e–01	4.34e–01	**0.00e+00**	**0.00e+00**	**1.39e+00**	**1.11e+00**	2.85e+00	1.19e+00
10	**7.96e–02**	**2.75e–01**	4.61e+00	2.34e+00	**1.72e+00**	**1.42e+00**	5.13e+01	1.24e+01
11	9.34e–01	9.00e–01	**2.57e–01**	**5.35e–01**	1.17e+01	3.14e+00	**3.56e+00**	**2.43e+00**
12	**2.93e+01**	**1.42e+02**	5.44e+02	7.14e+02	2.27e+05	1.11e+06	**1.79e+04**	**1.37e+04**
13	6.96e–01	1.50e–01	**4.84e–01**	**1.12e–01**	4.59e+00	5.15e–01	**3.19e+00**	**5.29e–01**
14	3.01e+00	3.49e–01	**1.94e+00**	**5.35e–01**	2.29e+01	5.78e–01	**2.11e+01**	**9.18e–01**

Table 26
Comparison with CoDE, SaDE, Ranked FIPS, and CMA-ES on CEC05 Functions – Results of 30D.

Bench.	CoDE		SaDE		Ranked FIPS		CMA-ES		GBSO	
Func.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00+e00**	**0.00+e00**	**0.00+e00**	**0.00+e00**	**0.00+e00**	**0.00+e00**	5.43e−09	9.80e−10	5.49e−14	1.04e−14
2	9.11e−06	1.11e−05	5.55e−10	6.99e−10	**0.00+e00**	**0.00+e00**	6.22e−09	8.95e−10	3.15e−04	2.10e−04
3	3.65e+05	1.60e+05	3.67e+05	1.82e+05	4.64e+05	1.29e+04	**5.55e−09**	**1.09e−09**	1.06e+06	3.61e+05
4	2.01e+01	5.61e+01	4.42e+01	1.25e+02	–	–	1.11e+04	3.02e+04	**1.83e−01**	**1.66e−01**
5	2.93e+04	2.21e−04	2.93e+04	1.09e+02	2.94e+03	2.90e+01	**9.62e−09**	**8.53e−09**	8.90e+01	7.94e+01
6	3.45e+00	3.67e+00	3.07e+01	3.24e+01	1.75e+01	1.68e−01	**5.90e−09**	**1.61e−09**	8.80e+01	1.47e+02
7	4.70e+03	1.03e−12	4.70e+03	4.15e−13	1.21e−02	7.70e−04	**5.31e−09**	**1.44e−09**	7.72e−03	7.91e−03
9	**1.34e−12**	**3.35e−12**	1.95e+00	1.27e+00	2.54e+01	6.61e−01	9.38e−01	1.18e+00	3.60e−01	6.30e−01
10	3.98e+01	1.55e+01	4.75e+01	1.13e+01	2.89e+01	5.65e−01	**1.65e+00**	**1.35e+00**	2.52e+01	7.85e+00
11	1.05e+01	4.09e+00	1.65e+01	2.85e+00	1.13e+01	2.87e−01	5.38e+00	3.13e+00	**1.35e+00**	**1.74e+00**
12	1.31e+05	2.54e+04	9.67e+04	1.47e+04	**4.52e+03**	**4.53e+02**	4.43e+04	2.15e+05	**4.44e+03**	**5.34e+03**
13	4.02e+00	1.22e+00	2.76e+00	3.60e−01	2.98e+00	7.1e−02	2.49e+00	5.13e−01	**1.73e+00**	**2.73e−01**
14	1.25e+01	3.82e−01	1.27e+01	1.83e−01	1.16e+01	3.7e−02	1.29e+01	4.19e−01	**1.01e+01**	**8.59e−01**

Table 27
Comparison with LSHADE on CEC14 Functions – Results of 10D.

Benchmark	LSHADE		GBSO	
Function	Mean	Std.	Mean	Std.
1	**0.00e+00**	**0.00e+00**	4.75e+04	4.85e+04
2	**0.00e+00**	**0.00e+00**	2.23e+03	2.15e+03
3	**0.00e+00**	**0.00e+00**	4.70e+02	7.70e+02
4	**2.94e+01**	**1.26e+01**	**2.61e+01**	**1.49e+01**
5	**1.41e+01**	**8.76e+00**	**1.57e+01**	**8.32e+00**
6	**1.75e−02**	**1.25e−01**	4.50e−01	8.71e−01
7	**3.04e−03**	**6.51e−03**	5.59e−02	2.93e−02
8	**0.00e+00**	**0.00e+00**	3.48e−02	1.61e−01
9	**2.34e+00**	**8.40e−01**	**2.95e+00**	**1.21e+00**
10	**8.57e−03**	**2.17e−02**	1.97e+00	2.40e+00
11	**3.21e+01**	**3.83e+01**	1.04e+02	8.01e+01
12	6.82e−02	1.92e−02	**8.37e−03**	**1.37e−02**
13	**5.16e−02**	**1.51e−02**	**5.05e−02**	**2.12e−02**
14	**8.14e−02**	**2.55e−02**	1.31e−01	6.04e−02
15	**3.66e−01**	**6.92e−02**	8.03e−01	2.54e−01
16	**1.24e+00**	**3.03e−01**	**1.20e+00**	**5.79e−01**
17	**9.77e−01**	**1.08e+00**	1.39e+03	1.66e+03
18	**2.44e−01**	**3.14e−01**	8.18e+03	6.48e+03
19	**7.73e−02**	**6.40e−02**	8.63e−01	3.31e−01
20	**1.85e−01**	**1.80e−01**	3.53e+02	8.12e+02
21	**4.08e−01**	**3.09e−01**	2.93e+02	3.18e+02
22	**4.41e−02**	**2.82e−02**	2.63e+01	4.49e+01
23	**3.29e+02**	**2.87e−13**	**3.29e+02**	**2.30e−13**
24	**1.07e+02**	**2.28e+00**	**1.07e+02**	**3.99e+00**
25	**1.33e+02**	**4.04e+01**	1.78e+02	3.17e+01
26	**1.00e+02**	**1.63e−02**	**1.00e+02**	**1.97e−02**
27	**5.81e+01**	**1.34e+02**	2.49e+02	1.14e+02
28	**3.81e+02**	**3.17e+01**	4.30e+02	6.51e+01
29	**2.22e+02**	**4.63e−01**	1.21e+05	4.11e+05
30	**4.65e+02**	**1.33e+01**	5.76e+02	1.16e+02

the post hoc procedures (Holm and Finner) are given in Table 24.

Results show that GBSO has the best performance on the simple multi-modal functions (f4–f16), which is closely followed by GABC. And although GABC is the best performer on the hybrid functions (f17–f22) and the composition functions (f23–f30) for D=10, the performance of GBSO on these functions is highly improved for larger problem sizes becoming the best performer in D=30 and D=50. In addition, results of the post hoc experiments show that GBSO is the best performer overall for the CEC14 benchmarks.

5.5. Comparison with the state-of-the-art algorithms

Table 25 presents a comparison between the proposed GBSO against CMA-ES [31] on the CEC05 benchmarks for D=10 and D=50. Table 26 adds CoDE, SaDE, and Ranked FIPS [32] to the comparison for D=30. Results of CoDE and SaDE are extracted from [13], results of Ranked FIPS are extracted from [32], and results of CMA-ES are extracted from [31]. All algorithms are run for $10000 \times D$ function evaluations. Note that in [32], Ranked FIPS was not applied to f4 and its entry in Table 26 is shown as '-'.

Results show that GBSO has a very competitive performance with CMA-ES across the different problem sizes providing a better solution for almost half of the functions under study. For D=30, GBSO provides the best solution on five out of the 13 tested functions as well as being the second best performer, shown in italic, on another five functions when compared against all other algorithms.

Table 27 presents a comparison between the proposed GBSO against LSHADE [33] on the CEC14 benchmarks for D=10. Tables 28, 29 add BLPSO [34] to the comparison for D=30 and D=50. BLPSO (Biogeography-based Learning PSO) is a recently proposed PSO variant that was shown to be better than FIPSO (The Fully Informed PSO) [35], CLPSO (Comprehensive Learning PSO) [36], and DMSPSO (Dynamic Multi-Swarm PSO) [37] as well as CMA-ES on the CEC14 benchmarks. Results of LSHADE and BLPSO are extracted from their respective references where all algorithms were run for $10000 \times D$ function evaluations. For each function, the best result is shown in bold while the second best result is shown in italic.

Results show that LSHADE is the best performer overall specially on the uni-modal functions (f1-f3) and the hybrid functions (f17-f22). GBSO offers some competitive results on the simple multi-modal functions (f4-f16) and the composition functions (f23-f30). In addition, results of the post hoc experiments, shown in Table 30, show that GBSO, although outperformed by LSHADE, is a better performer than BLPSO for the CEC14 benchmarks.

It's quite understandable that no algorithm will outperform all other competitors on all benchmark. And although GBSO is extremely competitive, with the state-of-the-art, on the CEC05 benchmarks and outperforms the recent BLPSO on the CEC14 benchmarks, it is still outperformed by LSHADE. It is worth noting that LSHADE uses some advanced mechanisms that are not adopted by GBSO including the success-history based adaptation of its parameters, the use of an external archive, and the linear reduction of the population size. All of these techniques are promising future directions for enhancing the performance of GBSO.

6. Conclusions

In this paper we introduced a Global-best BSO (GBSO) algorithm. The performance of BSO was improved using a combination of a fitness-based grouping approach, per-variable updates, following the global-best idea, and a re-initialization mechanism. The effect of each of these components on the original BSO algorithm was analyzed based

Table 28
Comparison with LSHADE and BLPSO on CEC14 Functions – Results of 30D.

Benchmark	LSHADE		BLPSO		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00e+00**	**0.00e+00**	2.99e+06	1.10e+06	8.38e+05	4.94e+05
2	**0.00e+00**	**0.00e+00**	5.09e+03	4.25e+03	8.55e+03	7.48e+03
3	**0.00e+00**	**0.00e+00**	3.67e+00	1.16e+01	2.22e+02	2.96e+02
4	**0.00e+00**	**0.00e+00**	2.68e+01	3.47e+01	6.15e+01	1.83e+01
5	**2.01e+01**	**3.68e−02**	2.08e+01	7.01e−02	2.03e+01	3.33e−01
6	**1.38e−07**	**9.89e−07**	9.37e−06	3.20e−05	9.84e−01	1.18e+00
7	**0.00e+00**	**0.00e+00**	9.47e−14	4.31e−14	9.17e−03	9.68e−03
8	**0.00e+00**	**0.00e+00**	2.32e−01	5.65e−01	1.70e−01	4.16e−01
9	**6.78e+00**	**1.48e+00**	3.54e+01	6.93e+00	2.31e+01	6.82e+00
10	**1.63e−02**	**1.58e−02**	8.83e+01	6.48e+01	3.80e+00	2.27e+00
11	1.23e+03	1.83e+02	2.08e+03	3.82e+02	**4.37e+02**	**2.56e+02**
12	1.61e−01	2.29e−02	8.83e−01	1.49e−01	**1.86e−02**	**2.07e−02**
13	**1.24e−01**	**1.75e−22**	2.21e−01	2.85e−02	1.57e−01	4.36e−02
14	2.42e−01	2.98e−02	**2.14e−01**	**2.88e−02**	2.21e−01	4.03e−02
15	**2.15e+00**	**2.51e−01**	7.41e+00	8.49e−01	3.38e+00	7.13e−01
16	**8.50e+00**	**4.58e−01**	9.67e+00	4.92e−01	**8.66e+00**	**7.46e−01**
17	**1.88e+02**	**7.50e+01**	1.86e+05	1.11e+05	7.28e+04	4.65e+04
18	**5.91e+00**	**2.89e+00**	9.05e+02	1.20e+03	2.39e+03	3.24e+03
19	**3.68e+00**	**6.80e−01**	**3.74e+00**	**6.12e−01**	4.72e+00	1.02e+00
20	**3.08e+00**	**1.47e+00**	3.12e+02	3.48e+02	8.80e+01	3.10e+01
21	**8.68e+01**	**8.99e+01**	3.85e+04	3.19e+04	2.32e+04	1.53e+04
22	**2.76e+01**	**1.79e+01**	1.16e+02	6.86e+01	2.24e+02	9.10e+01
23	**3.15e+02**	**4.02e−13**	**3.15e+02**	**6.11e−13**	**3.15e+02**	**1.55e−04**
24	2.24e+02	1.06e+00	2.22e+02	7.39e−01	**2.00e+02**	**5.98e−02**
25	**2.03e+02**	**4.96e−02**	2.05e+02	4.41e−01	**2.03e+02**	**3.05e−01**
26	**1.00e+02**	**1.55e−02**	1.04e+02	1.82e+01	**1.00e+02**	**3.90e−02**
27	**3.00e+02**	**2.40e−13**	3.08e+02	2.94e+01	3.96e+02	7.11e+01
28	8.40e+02	1.40e+01	**7.87e+02**	**5.22e+01**	**7.94e+02**	**7.80e+01**
29	**7.17e+02**	**5.13e+00**	1.39e+03	1.39e+02	1.71e+05	1.21e+06
30	1.25e+03	6.20e+02	**1.19e+03**	**2.49e+02**	2.03e+03	6.51e+02

Table 29
Comparison with LSHADE and BLPSO on CEC14 Functions – Results of 50D.

Benchmark	LSHADE		BLPSO		GBSO	
Function	Mean	Std.	Mean	Std.	Mean	Std.
1	**1.24e+03**	**1.52e+03**	5.10e+06	1.28e+06	2.22e+06	7.45e+05
2	**0.00e+00**	**0.00e+00**	3.44e+03	2.35e+03	9.26e+03	8.21e+03
3	**0.00e+00**	**0.00e+00**	4.23e+01	8.97e+01	1.59e+03	8.57e+02
4	**5.89e+01**	**4.56e+01**	8.64e+01	5.04e+00	9.66e+01	2.90e+00
5	**2.02e+01**	**4.59e−02**	2.09e+01	5.07e−02	2.09e+01	3.73e−01
6	2.64e−01	5.23e−01	**9.22e−02**	**3.21e−01**	2.08e+00	1.65e+00
7	**0.00e+00**	**0.00e+00**	2.92e−13	6.46e−14	3.04e−03	4.34e−03
8	**2.58e−09**	**7.48e−09**	4.97e−01	8.16e−01	1.47e+00	1.30e+00
9	**1.14e+01**	**2.13e+00**	7.10e+01	9.02e+00	4.76e+01	1.06e+01
10	**1.22e−01**	**4.13e−02**	3.63e+02	1.81e+02	6.76e+00	6.01e+00
11	3.22e+03	3.30e+02	4.46e+03	4.77e+02	**1.58e+03**	**5.25e+02**
12	2.19e−01	2.82e−02	8.77e−01	1.18e−01	**1.80e−02**	**1.30e−02**
13	**1.60e−01**	**1.83e−02**	2.86e−01	3.72e−02	2.56e−01	4.56e−02
14	2.97e−01	2.47e−02	2.65e−01	2.42e−02	**2.36e−01**	**6.71e−02**
15	**5.15e+00**	**5.08e−01**	1.48e+01	1.33e+00	6.36e+00	1.25e+00
16	**1.69e+01**	**4.81e−01**	1.82e+01	4.73e−01	**1.71e+01**	**8.47e−01**
17	**1.40e+03**	**5.13e+02**	5.97e+05	2.10e+05	1.89e+05	1.51e+05
18	**9.73e+01**	**1.38e+01**	3.73e+02	3.68e+02	1.51e+03	1.45e+03
19	**8.30e+00**	**1.81e+00**	2.16e+01	9.78e+00	1.15e+01	1.37e+00
20	**1.39e+01**	**4.56e+00**	2.57e+02	1.41e+02	1.84e+02	4.61e+01
21	**5.15e+02**	**1.49e+02**	3.80e+05	1.44e+05	1.44e+05	7.87e+04
22	**1.14e+02**	**7.50e+01**	2.64e+02	1.30e+02	2.74e+02	1.40e+02
23	**3.44e+02**	**4.44e−13**	**3.44e+02**	**2.56e−13**	**3.44e+02**	**1.60e−03**
24	2.75e+02	6.62e−01	**2.58e+02**	**4.07e+00**	**2.56e+02**	**2.51e+00**
25	**2.05e+02**	**3.65e−01**	2.10e+02	7.36e−01	2.07e+02	8.45e−01
26	**1.00e+02**	**7.85e−02**	1.47e+02	5.08e+01	1.32e+02	4.68e+01
27	**3.33e+02**	**3.03e+01**	**3.24e+02**	**2.85e+01**	4.35e+02	5.76e+01
28	1.11e+03	2.91e+01	1.14e+03	4.20e+01	**1.07e+03**	**7.14e+01**
29	**7.95e+02**	**2.40e+01**	1.36e+03	1.82e+02	2.46e+03	6.22e+02
30	**8.66e+03**	**4.13e+02**	9.09e+03	3.05e+02	9.11e+03	6.40e+02

Table 30
Statistical Tests Results For LSHADE, BLPSO, and GBSO – CEC14 Functions.

Algorithm	Friedman Ranking	Holm p-value	Finner p-value
BLPSO	2.41	0.025	0.025
GBSO	2.23	0.05	0.05
LSHADE	**1.37**	–	–

on the final solution reached and the maintained population's diversity. In addition, experiments were conducted to assess the algorithm's sensitivity to its global-best parameters. Moreover, the performance of GBSO was compared against three BSO variants as well as the global-best versions of other nature-inspired algorithms on a wide range of benchmark libraries.

Experiments proved the superior performance of GBSO compared to previous BSO improved algorithms. Further experiments demonstrated the highly competitive performance against other popular global-best heuristic algorithms. Although GBSO had the slowest speed of convergence on the CEC05 benchmarks, it was proved to be the best performer overall. In addition, it showed a very desirable property by having an improved performance with higher dimensions on the hybrid and composition functions of the CEC14 benchmarks. Finally, GBSO was shown to have a very competitive performance with a number of the state-of-the-art algorithms on the CEC05 benchmarks. On the CEC14 benchmarks, GBSO produced better results than a recent PSO algorithm but is still outperformed by LSHADE.

Future research directions involve proposing a mechanism to adapt the algorithm's parameters, experimenting with different re-initialization schemes, applying GBSO in several engineering optimization problems, using GBSO within a cooperative co-evolutionary framework, and developing a discrete GBSO version.

References

[1] J. Kennedy, R. C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[2] R. C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[3] M. Dorigo, Optimization, learning and natural algorithms (in italian), (Ph.D. thesis), Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
[4] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Rep. TR06, Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.
[5] X. S. Yang, S. Deb, Cuckoo search via lévy flights, in: Proceedings of World Congress on Nature and Biologically Inspired Computing, 2009, pp. 210–214.
[6] X. S. Yang, A new metaheuristic bat-inspired algorithm, in: Proceedings of Nature Inspired Cooperative Strategies for Optimization, 2010, pp. 65–74.
[7] Y. Shi, Brain storm optimization algorithm, in: Proceedings of International Conference on Swarm Intelligence, 2011, pp. 303–309.
[8] Y. Shi, An optimization algorithm based on brainstorming process, Int. J. Swarm Intell. Res. 2 (4) (2011) 35–62.
[9] Z. Zhan, J. Zhang, Y. Shi, H. Liu, A modified brain storm optimization algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, 2012, pp. 1969–1976.
[10] Z. Cao, Y. Shi, X. Rong, B. Liu, Z. Du, B. Yang, Random grouping brain storm optimization algorithm with a new dynamically changing step size, in: Proceedings International Conference on Swarm Intelligence, 2015, pp. 387–364.
[11] J. Chen, J. Wang, S. Cheng, Y. Shi, Brain storm optimization with agglomerative hierarchical clustering analysis, in: Proceedings of the 7th International Conference on Swarm Intelligence, ICSI, 2016, pp. 115–122.
[12] D. Zhou, Y. Shi, S. Cheng, Brain storm optimization algorithm with modified step-size and individual generation, in: Proceedings International Conference on Swarm Intelligence, 2012, pp. 243–252.
[13] Z. Cao, X. Hei, L. Wang, Y. Shi, X. Rong, An improved brain storm optimization with differential evolution strategy for applications of ANNS, Math. Probl. Eng. 2015 (2015) 1–18.
[14] S. Cheng, Y. Shi, Q. Qin, Q. Zhang, R. Bai, Population diversity maintenance in brain storm optimization algorithm, J. Artifical Intellgence Soft Comput. Res. 4 (2) (2014) 83–97.
[15] S. Cheng, Y. Shi, Q. Qin, T. O. Ting, R. Bai, Maintaining population diversity in brain storm optimization algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, 2014, pp. 3230–3237.
[16] M. El-Abd, Brain storm optimization algorithm with re-initialized ideas and modified step size, in: Proceedings of IEEE Congress on Evolutionary Computation, 2016, pp. 2682–2686.
[17] Standard pso 2011. ⟨http://www.particleswarm.info⟨.
[18] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (7) (2010) 3166–3173.
[19] M. El-Abd, An improved global-best harmony search algorithm, Appl. Math. Comput. 222 (2013) 94–106.
[20] 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, Tech. Rep. 2005005, ITT Kanpur, India, 2005.
[21] J. J. Liang, B.-Y. Qu, P. N. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Tech. rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2014.⟨http://www.ntu.edu.sg/home/epnsugan/⟩.
[22] S. Cheng, Q. Qin, J. Chen, Y. Shi, Brain storm optimization algorithm: a review, Artif. Intell. Rev. (2016) 1–14.
[23] J. Chen, S. Cheng, Y. Chen, Y. Xie, Y. Shi, Enhanced brain storm optimization algorithm for wireless sensor networks deployment, in: Proceedings of the International Conference on Swarm Intelligence, 2015, pp. 373–381.
[24] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evolut. Comput. 15 (1) (2011) 55–66.
[25] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[26] B. Basturk, D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2006.
[27] B.Y. Qu, J.J. Liang, Z.Y. Wang, Q. Chen, P.N. Suganthan, Novel benchmark functions for continuous multimodal optimization with comparative resultsSwarm Evolut. Comput. 26 (2016) 23–34.
[28] J. Derrac, S. Garcia, 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 Evolut. Comput. 1 (1) (2011) 3–18.
[29] Keel software, ⟨http://sci2s.ugr.es/keel/download.php⟩, 2012.
[30] M. El-Abd, Performance assessment of foraging algorithms vs. evolutionary algorithms, Inf. Sci. 182 (1) (2012) 243–263.
[31] A. Auger, N. Hansen, A restart cma evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1769–1776.
[32] J. Jordan, S. Helwig, R. Wanka, Social interaction in particle swarm optimization, the ranked fips, and adaptive multi-swarms, in: Proceedings of Genetic and Evolutionary Computation Conference GECCO, 2008, pp. 49–56.
[33] R. Tanabe, A. S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2014, pp. 1658–1665.
[34] X. Chen, H. Tianfield, C. Mei, W. Du, G. Liu, Biogeography-based learning particle swarm optimization, Soft Computing, Doi:10.1007/s00500-016-2307-7.
[35] R. Mendes, R. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evolut. Comput. 8 (3) (2004) 204–210.
[36] J. Liang, A.K. Qin, P.N. Suganthan, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (3) (2006) 281–295.
[37] J. Liang, P. N. Suganthan, Improving the search performance of shade using linear population size reduction, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2005, pp. 124–129.