Applied Mathematics and Computation 219 (2013) 8121–8144

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

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

Backtracking Search Optimization Algorithm for numerical optimization problems

Pinar Civicioglu

Erciyes University, College of Aviation, Dept. of Aircraft Electrics and Electronics, Kayseri, Turkey

ARTICLE INFO

Keywords:
Swarm intelligence
Evolutionary algorithms
Differential evolution
Numerical optimization
Wilcoxon Signed-Rank Test

ABSTRACT

This paper introduces the Backtracking Search Optimization Algorithm (BSA), a new evolutionary algorithm (EA) for solving real-valued numerical optimization problems. EAs are popular stochastic search algorithms that are widely used to solve non-linear, non-differentiable and complex numerical optimization problems. Current research aims at mitigating the effects of problems that are frequently encountered in EAs, such as excessive sensitivity to control parameters, premature convergence and slow computation. In this vein, development of BSA was motivated by studies that attempt to develop simpler and more effective search algorithms. Unlike many search algorithms, BSA has a single control parameter. Moreover, BSA's problem-solving performance is not over sensitive to the initial value of this parameter. BSA has a simple structure that is effective, fast and capable of solving multimodal problems and that enables it to easily adapt to different numerical optimization problems. BSA's strategy for generating a trial population includes two new crossover and mutation operators. BSA's strategies for generating trial populations and controlling the amplitude of the search-direction matrix and search-space boundaries give it very powerful exploration and exploitation capabilities. In particular, BSA possesses a memory in which it stores a population from a randomly chosen previous generation for use in generating the search-direction matrix. Thus, BSA's memory allows it to take advantage of experiences gained from previous generations when it generates a trial preparation. This paper uses the Wilcoxon Signed-Rank Test to statistically compare BSA's effectiveness in solving numerical optimization problems with the performances of six widely used EA algorithms: PSO, CMAES, ABC, JDE, CLPSO and SADE. The comparison, which uses 75 boundary-constrained benchmark problems and three constrained real-world benchmark problems, shows that in general, BSA can solve the benchmark problems more successfully than the comparison algorithms.

1. Introduction

Optimization is a very important research area in applied mathematics [1–5]. Optimization algorithms aim to find the best values for a system's parameters under various conditions. The first step in solving an optimization problem is determining the objective function that states the relations between the system parameters and system constraints. For various reasons, the objective function may have a non-linear, complex or non-differentiable form. Optimization problems are usually designed in a way that defines the global optimum of an objective function as the global minimum. The process of searching for the global optimum for an optimization problem is called global optimization. Desirable

E-mail address: civici@erciyes.edu.tr

0096-3003/$ - see front matter
http://dx.doi.org/10.1016/j.amc.2013.02.017

features for optimization algorithms include the ability to reach a problem's global minimum value quickly with a small number of control parameters and low computational cost, as well as robustness and ease of application to different problem models [6–8]. Since the size of the search space increases with the dimension of the optimization problem, it is difficult to find the global optimum for such problems with classical techniques [9]. When the objective function for an optimization problem is non-linear and non-differentiable, evolutionary algorithm (EA) techniques are typically used to find the global optimum [10–13]. EAs have been used for mechanical design problems [14], communication applications [15], image processing applications [16], speech recognition problems [17], sensor deployment problems [18], data mining applications [19], IIR filter design [20], solar radiation modeling [21], lens design [22], dynamic analysis of chemical processes [23], phase stability and equilibrium calculations for reactive systems [24], batch scheduling for polypropylene processes [25], dynamic optimization of biochemical processes [26] and many other engineering problems.

The most commonly used EA optimization techniques are based on swarm intelligence [12,14,20,27–33] and genetic evolution [1–7,34,35]. Swarm-intelligence optimization algorithms generally use simplified mathematical models of complex social behaviors of living creatures. However, recent swarm-intelligence based optimization algorithms have also been inspired by natural events involving gravitational physics [31], behavior of birds [29] and various branches of art such as music [36].

Unlike classical optimization techniques, EAs do not guarantee finding the optimum parameter values for a problem. However, also unlike classical optimization algorithms, EAs are sufficiently flexible to solve different types of problems. EAs must have global exploration and local exploitation abilities [12,31]. A global exploration ability means that the optimization algorithm effectively uses the entire search space, while a local exploitation ability means that the optimization algorithm searches for the best solution near a new solution it has already discovered. EAs generally use their exploration ability to acquire the new solutions that are needed to avoid local minimums in their first iterations. As the iterations progress, an EA's exploitation ability has a stronger influence on the solutions that are generated. The success of an optimization algorithm significantly depends on its exploration and exploitation abilities and the natural balance between them. Since new and complex numerical optimization problems that are difficult to solve with conventional methods are frequently encountered in scientific research, there is an ongoing need to develop new optimization algorithms based on different techniques [12,36–40].

Nature-inspired analogical models have been used to design many EAs. The ant colony optimization algorithm is based on ants' strategy for accessing food sources [27]. The biogeography-based optimization algorithm models the bio-interactions among living creatures in a habitat [28]. The cuckoo search algorithm models parasitic bio-interactions in living creatures [29,30]. The gravitational search algorithm models the interactions of objects arising from universal gravitation laws [31]. The particle swarm optimization algorithm (PSO) simulates the choreographic movements of superorganisms such as flocks of birds and schools of fish moving in concert [24,25,38–42]. The artificial bee colony algorithm (ABC) is inspired by honeybees' food-searching behavior [12,13,30]. These analogical models are also useful for describing the EAs to researchers familiar with the widely known concepts.

In an EA, protection of a population's genetic diversity is very important for the population's ability to iteratively sustain its development. It is generally believed that in nature, the genetic diversity of a population results from basic genetic processes such as recombination, crossover, mutation, selection and adaptation [1,11,12]. Many EAs are based on basic genetic rules, such as ABC, the cuckoo search, the covariance matrix adaptation evolution strategy (CMAES) [37], genetic algorithms [1,2] and the differential evolution algorithm (DE) [30,34,35].

DE, a genetic algorithm, has a very simple algorithmic structure that can easily adapt to different types of problems, and it has been used to solve many engineering problems. The standard DE has five mutation strategies and two crossover strategies [7,10,11,30,34,35]. DE's search for the global minimum is sensitive to the mutation and crossover strategy it uses, initial values of the mutation and crossover factors, size of its population and the number of iterations. Advanced versions of DE, such as the adaptive differential evolution algorithm (JDE) [6], the parameter adaptive differential evolution algorithm (JADE) [10] and the self-adaptive differential evolution algorithm (SADE) [35], were developed to improve DE's problem-solving success rate. In contrast, ABC is an EA that differs from DE by virtue of using genetic rules but no crossover strategy [12,13]. Moreover, when searching for the global minimizer, ABC tends to make more use of solutions in the search space that provide better fitness values. The comprehensive learning particle swarm optimizer (CLPSO) [8] and PSO2011 [42] are advanced versions of the standard PSO [40,41]. PSO2011 incorporates many improvements of PSO that have been identified by years of studies. CMAES, which is widely used in scientific applications, is based on an evolutionary development of the covariance matrix [27].

EAs are population-based stochastic search mechanisms that search for near-optimal solutions to a problem. An EA tries to evolve an individual into one with a better fitness value through a 'trial individual'. To generate a trial individual, the EA chooses existing individuals as raw genetic material and combines these using various genetic operators. If the trial individual has a better fitness value than the original individual, the trial individual replaces it in the next-generation population. EAs radically differ from one another based on their strategies for generating trial individuals. Because these strategies have a considerable effect on their problem-solving success and speed, ongoing efforts are aimed at developing EAs with faster and more successful problem-solving processes.

The algorithm proposed in this paper, BSA, is a new EA. BSA's unique mechanism for generating a trial individual enables it to solve numerical optimization problems successfully and rapidly. BSA uses three basic genetic operators –

selection, mutation and crossover – to generate trial individuals. BSA has a random mutation strategy that uses only one direction individual for each target individual, in contrast with many genetic algorithms such as DE and its derivatives JDE, JADE and SADE. BSA randomly chooses the direction individual from individuals of a randomly chosen previous generation. BSA uses a non-uniform crossover strategy that is more complex than the crossover strategies used in many genetic algorithms.

Since the PSO, CMAES, ABC, JDE, CLPSO and SADE algorithms are widely used in scientific applications, these algorithms have been selected as comparison algorithms for evaluating BSA's success in solving numerical optimization problems.

This paper uses three test sets to examine the success of BSA and the comparison algorithms in solving numerical optimization problems. The first test set includes 50 widely used standard benchmark problems [12,13], the second test set includes 25 benchmark problems used in CEC2005 [43] and the third test set includes three real-world problems used in CEC2011 [44]: the Circular Antenna Array Design problem (Antenna), the Spread Spectrum Radar Polly Phase Code Design problem (Radar) and the Parameter Estimation for Frequency-Modulated Sound Waves problem (FM).

The rest of this paper is organized as follows. Section 2 introduces Backtracking Search Optimization Algorithm, Section 3 describes the experiments and Section 4 presents the conclusions.

2. Backtracking Search Optimization Algorithm (BSA)

BSA is a population-based iterative EA designed to be a global minimizer. BSA can be explained by dividing its functions into five processes as is done in other EAs: initialization, selection-I, mutation, crossover and selection-II.

Algorithm 1 presents BSA's general structure.

Algorithm 1. General Structure of BSA

```
1. Initialization
repeat
|  2. Selection-I
|  Generation of Trial-Population
|  |  3. Mutation
|  |  4. Crossover
|  end
|  5. Selection-II
until stopping conditions are met;
```

2.1. Initialization

BSA initializes the population P with Eq. (1):

$$P_{i,j} \sim \mathrm{U}(low_j, up_j) \tag{1}$$

for $i = 1,2,3,\ldots,N$ and $j = 1,2,3,\ldots,D$, where N and D are the population size and the problem dimension, respectively, U is the uniform distribution and each P_i is a target individual in the population P.

2.2. Selection-I

BSA's Selection-I stage determines the historical population *oldP* to be used for calculating the search direction. The initial historical population is determined using Eq. (2):

$$oldP_{i,j} \sim \mathrm{U}(\mathrm{low_j}, \mathrm{up_j}). \tag{2}$$

BSA has the option of redefining *oldP* at the beginning of each iteration through the 'if-then' rule in Eq. (3):

$$if\ \ a < b\ then\ oldP := P | a, b \sim U(0,1), \tag{3}$$

where $:=$ is the update operation. Eq. (3) ensures that BSA designates a population belonging to a randomly selected previous generation as the historical population and remembers this historical population until it is changed. Thus, BSA has a memory.

After *oldP* is determined, Eq. (4) is used to randomly change the order of the individuals in *oldP*:

$$oldP := permuting(oldP). \tag{4}$$

The permuting function used in Eq. (4) is a random shuffling function.

2.3. Mutation

BSA's mutation process generates the initial form of the trial population *Mutant* using Eq. (5).

$$Mutant = P + \mathrm{F} \cdot (oldP - P). \tag{5}$$

In Eq. (5), F controls the amplitude of the search-direction matrix ($oldP - P$). Because the historical population is used in the calculation of the search-direction matrix, BSA generates a trial population, taking partial advantage of its experiences from previous generations. This paper uses the value $\mathrm{F} = 3 \cdot rndn$, where $rndn \sim N(0,1)$ (N is the standard normal distribution).

2.4. Crossover

BSA's crossover process generates the final form of the trial population T. The initial value of the trial population is *Mutant*, as set in the mutation process. Trial individuals with better fitness values for the optimization problem are used to evolve the target population individuals. BSA's crossover process has two steps. The first step calculates a binary integer-valued matrix (*map*) of size $N \cdot D$ that indicates the individuals of T to be manipulated by using the relevant individuals of P. If $map_{n,m} = 1$, where $n \in \{1,2,3,\ldots,N\}$ and $m \in \{1,2,3,\ldots,D\}$, T is updated with $T_{n,m} := P_{n,m}$.

Algorithm 2 shows BSA's unique crossover strategy.

Algorithm 2. Crossover Strategy of BSA

```
   Input: Mutant, mixrate, N and D.
   Output: T: Trial-Population.
 0 map(1:N,1:D) = 1   // Initial-map is an N-by-D matrix of ones.
 1 if a < b | a,b ~ U(0,1) then
 2 |  for i from 1 to N do
 3 |  |  map_{i,u(1:⌈mixrate·rnd·D⌉)} = 0 | u = permuting(⟨1,2,3,...,D⟩)
 4 |  end
 5 else
 6 |  for i from 1 to N do, map_{i,randi(D)} = 0, end
 7 end
 8 T := Mutant   // Initial T
 9 for i from 1 to N do
10 |  for j from 1 to D do
11 |  |  if map_{i,j} = 1 then T_{i,j} := P_{i,j}
12 |  end
13 end
```

In Algorithm-2, $\lceil\rceil$ (on line 3) indicates the ceiling function, defined as $rnd \sim U(0,1)$. BSA's crossover strategy is quite different from the crossover strategies used in DE and its variants. The mix rate parameter (*mixrate*) in BSA's crossover process controls the number of elements of individuals that will mutate in a trial by using $\lceil mixrate \cdot rnd \cdot D \rceil$ (Algorithm 2, line 3). The function of the mix rate is quite different from the crossover rate used in DE.

Two predefined strategies are randomly used to define BSA's *map*. The first strategy uses *mixrate* (Algorithm 2, lines 2–4). The second strategy allows only one randomly chosen individual to mutate in each trial (Algorithm 2, line 6). BSA's crossover process is more complex than the processes used in DE.

Some individuals of the trial population obtained at the end of BSA's crossover process can overflow the allowed search-space limits as a result of BSA's mutation strategy. The individuals beyond the search-space limits are regenerated using Algorithm 3.

Algorithm 3. Boundary Control Mechanism of BSA

```
Input: T, Search space limits (i.e., low_j, up_j)
Output: T
for i from 1 to N do
|  for j from 1 to D do
|  |  if (T_{i,j} < low_j) or (T_{i,j} > up_j) then
|  |  |  T_{i,j} = rnd · (up_j − low_j) + low_j
|  |  end
|  end
end
```

2.5. Selection-II

In BSA's Selection-II stage, the T_is that have better fitness values than the corresponding P_is are used to update the P_is based on a greedy selection. If the best individual of P (P_{best}) has a better fitness value than the global minimum value obtained so far by BSA, the global minimizer is updated to be P_{best}, and the global minimum value is updated to be the fitness value of P_{best}. The structure of BSA is quite simple; thus, it is easily adapted to different numerical optimization problems.

Algorithm 4 presents pseudo code for BSA:

Algorithm 4. Pseudo-Code of BSA

```
Input: ObjFun, N, D, maxcycle, mixrate, low_{1:D}, up_{1:D}
Output: globalminimum, globalminimizer
// rnd ~ U(0,1), rndn ~ N(0,1), w = rndint(·), rndint(·) ~ U(1,·) | w ∈ {1,2,3,...,·}
function bsa(ObjFun, N, D, maxcycle, low, up)
// INITIALIZATION
globalminimum = inf
for i from 1 to N do
    for j from 1 to D do
        P_{i,j} = rnd · (up_j − low_j) + low_j // Initialization of population, P.
        oldP_{i,j} = rnd · (up_j − low_j) + low_j // Initialization of oldP.
    end
    fitnessP_i = ObjFun(P_i) // Initial-fitness values of P
end
for iteration from 1 to maxcycle do
    // SELECTION-I
    if (a < b | a, b ~ U(0,1)) then oldP := P end
    oldP := permuting(oldP) // 'permuting' arbitrary changes in positions of two
    individuals in oldP.
    Generation of Trial-Population
        // MUTATION
        mutant = P + 3 · rndn · (oldP − P)
        // CROSSOVER
        map_{1:N,1:D} = 1 // Initial-map is an N-by-D matrix of ones.
        if (c < d | c, d ~ U(0,1)) then
            for i from 1 to N do
                map_{i,u(1:⌈mixrate·rnd·D⌉)} = 0 | u = permuting(⟨1,2,3,...,D⟩)
            end
        else
            for i from 1 to N do, map_{i,randi(D)} = 0, end
        end
        // Generation of Trial Population, T
        T := mutant
        for i from 1 to N do
            for j from 1 to D do
                if map_{i,j} = 1 then T_{i,j} := P_{i,j}
            end
        end
        // Boundary Control Mechanism
        for i from 1 to N do
            for j from 1 to D do
                if (T_{i,j} < low_j) or (T_{i,j} > up_j) then
                    T_{i,j} = rnd · (up_j − low_j) + low_j
                end
            end
        end
    end
    // SELECTION-II
    fitnessT = ObjFnc(T)
    for i from 1 to N do
        if fitnessT_i < fitnessP_i then
            fitnessP_i := fitnessT_i
            P_i := T_i
        end
    end
    fitnessP_best = min(fitnessP) | best ∈ {1,2,3,...,N}
    if fitnessP_best < globalminimum then
        globalminimum := fitnessP_best
        globalminimizer := P_best
        // Export globalminimum and globalminimizer
    end
end
```

The software codes in Matlab of BSA can be found at [45].

2.6. Comparison of BSA with the comparison algorithms

- BSA is an EA, similar to the comparison algorithms PSO, CMAES, ABC, JDE, CLPSO and SADE.
- BSA's mutation and crossover mechanisms are different from those of DE and its advanced versions JDE, JADE and SADE. BSA's mutation mechanism uses only one individual from a previous population, and BSA's crossover mechanism is more complex than the crossover mechanisms of DE and its advanced versions.
- Unlike ABC, JDE and SADE, BSA has no tendency to use individuals in the population with better fitness values more frequently than others. This makes BSA more successful in solving multimodal problems.
- BSA has a much simpler structure than the comparison algorithms.
- BSA's mutation and crossover strategies are radically different from those of DE and its advanced versions.
- BSA's boundary control mechanism is different from those of ABC and DE and its advanced versions.
- BSA remembers the population of a randomly selected generation for use in calculating the search-direction matrix. PSO, CMAES, ABC, JDE, CLPSO and SADE do not use previous generation populations.
- BSA is a dual-population algorithm that uses both the current and historical populations, unlike the comparison algorithms.

3. Experiments

This section presents in detail the tests and benchmark problems, statistical analysis, arithmetic precision and control parameters and stopping conditions used for the optimization algorithms in the tests, along with the statistical results.

3.1. Tests and benchmark problems

Three tests were conducted to examine the relative success of BSA and the comparison algorithms in solving the numerical optimization problems.

Test 1 involved 50 widely used benchmark problems. Detailed information about these problems is provided in [12,13]. Table 1 summarizes several features of the benchmark problems used in Test 1.

Test 2 involved 25 benchmark problems used in CEC2005. Detailed information about these problems is provided in [43]. Table 2 summarizes several features of the benchmark problems used in Test 2.

Test 3 involved three of the real-world problems used in CEC2011: the Circular Antenna Array Design Problem (Antenna), the Spread Spectrum Radar Polly Phase Code Design (Radar) and Parameter Estimation for Frequency-Modulated Sound Waves (FM). Detailed definitions of these problems are provided in [44]. Table 3 summarizes the definitions of the real-world problems used in Test 3.

3.2. Control parameters for the optimization algorithms used in the tests

Table 4 gives the initial values of the relevant control parameters for the EAs tested in this paper.

The relevant common control parameters for the evolutionary searches are as follows:

- The maximum number of times the objective function is evaluated is 2,000,000.
- The population size is 30.
- CMAES requires using a population size that varies with the problem size. Therefore, the population size (N_{CMAES}) for CMAES was calculated using Eq. (6):

$$N_{CMAES} = 4 + \lfloor 3 \cdot \log\,(Dimension\ of\ Problem) \rfloor. \tag{6}$$

3.3. Stopping conditions for the optimization algorithms used in the tests

The predetermined criteria used to stop the algorithms' searches are as follows.

- If the absolute value of the objective function is less than 10^{-16}, stop.
- If the algorithm has failed to find a better solution than the existing solution during the last 200,000 function evaluations, stop.
- If the number of function evaluations reaches 2,000,000, stop.
- If the maximum number of generations has been reached, stop.

In the tests, the benchmark problems were solved 30 times, each time using a different initial population. In each test, the evolutionary computing algorithms used the same initial population. The global minimum and runtime values for each test were recorded for detailed statistical analysis. All tests and statistical analyses were conducted using Matlab. The simple statistical values acquired during the tests provide information about the problem-solving abilities of the tested algorithms.

3.4. Simple numerical example showing the functions of BSA

In this section, the benchmark problem F43 is used to show in detail function of BSA for a small numerical example. F43 has two variables, and the population size in this example is predefined to be 3. Therefore, the size of P is 3×2. Eq. (7) defines the objective function for F43:

$$ObjFun(x) = 4 \cdot x_1^2 + 2.1 \cdot x_1^4 + \frac{1}{3} \cdot x_1^6 + x_1 \cdot x_2 - 4 \cdot x_2^2 + 4 \cdot x_2^4. \tag{7}$$

The search-space limits are defined to be $-5 \leqslant x_1, x_2 \leqslant 5$. Table 5 shows the values BSA acquired for various variables during its first five iterations while solving F43.

Table 1
The benchmark problems used in Test 1 (Dim: Dimension, Low, Up: Limits of search space, M: Multimodal, N: Non-Separable, U: Unimodal, S: Separable).

Problem	Name	Type	Low	Up	Dim
F1	Foxholes	MS	−65.536	65.536	2
F2	GoldsteinPrice	MN	−2	2	2
F3	Penalized	MN	−50	50	30
F4	Penalized2	MN	−50	50	30
F5	Ackley	MN	−32	32	30
F6	Beale	UN	−4.5	4.5	5
F7	Bohachecsky1	MS	−100	100	2
F8	Bohachecsky2	MN	−100	100	2
F9	Bohachecsky3	MN	−100	100	2
F10	Booth	MS	−10	10	2
F11	Branin	MS	−5	10	2
F12	Colville	UN	−10	10	4
F13	DixonPrice	UN	−10	10	30
F14	Easom	UN	−100	100	2
F15	Fletcher	MN	−3.1416	3.1416	2
F16	Fletcher	MN	−3.1416	3.1416	5
F17	Fletcher	MN	−3.1416	3.1416	10
F18	Griewank	MN	−600	600	30
F19	Hartman3	MN	0	1	3
F20	Hartman6	MN	0	1	6
F21	Kowalik	MN	−5	5	4
F22	Langermann	MN	0	10	2
F23	Langermann	MN	0	10	5
F24	Langermann	MN	0	10	10
F25	Matyas	UN	−10	10	2
F26	Michalewics	MS	0	3.1416	2
F27	Michalewics	MS	0	3.1416	5
F28	Michalewics	MS	0	3.1416	10
F29	Perm	MN	−4	4	4
F30	Powell	UN	−4	5	24
F31	Powersum	MN	0	4	4
F32	Quartic	US	−1.28	1.28	30
F33	Rastrigin	MS	−5.12	5.12	30
F34	Rosenbrock	UN	−30	30	30
F35	Schaffer	MN	−100	100	2
F36	Schwefel	MS	−500	500	30
F37	Schwefel_1_2	UN	−100	100	30
F38	Schwefel_2_22	UN	−10	10	30
F39	Shekel10	MN	0	10	4
F40	Shekel5	MN	0	10	4
F41	Shekel7	MN	0	10	4
F42	Shubert	MN	−10	10	2
F43	Sixhumpcamelback	MN	−5	5	2
F44	Sphere2	US	−100	100	30
F45	Step2	US	−100	100	30
F46	Stepint	US	−5.12	5.12	5
F47	Sumsquares	US	−10	10	30
F48	Trid	UN	−36	36	6
F49	Trid	UN	−100	100	10
F50	Zakharov	UN	−5	10	10

3.5. Statistical analysis

Owing to their stochastic nature, EAs may arrive at better or worse solutions than solutions they have previously reached by chance during their search for new solutions to a problem. Because of such cases, it is beneficial to use statistical tools to compare the problem-solving success of one EA with that of another.

The simple statistical parameters that can be derived from the results of an algorithm solving a specific numerical problem K times under different initial conditions – i.e. the mean solution (*mean*), the standard deviation of the mean solution (*std*) and the best solution (*best*) – only provide information about the algorithm's behavior in solving that particular problem.

3.5.1. Statistical pairwise test tool: the Wilcoxon Signed-Rank Test

For pairwise comparison of the problem-solving success of EAs, a problem-based or multi-problem-based statistical comparison method can be used [46]. A problem-based comparison can use the global minimum values obtained for the problem as the result of several runs. Problem-based pairwise comparisons are widely used to determine which of two algorithms

Table 2
The benchmark problems used in Test 2 (Dim: Dimension, Low, Up: Limits of search space, M: Multimodal, E: Expanded, H: Hybrid, C: Composition, U: Unimodal).

Problem	Name	Type	Dim	Low	Up
F51	Shifted sphere	U	10	−100	100
F52	Shifted Schwefel	U	10	−100	100
F53	Shifted rotated high conditioned elliptic function	U	10	−100	100
F54	Shifted Schwefels problem 1.2 With noise	U	10	−100	100
F55	Schwefels problem 2.6	U	10	−100	100
F56	Shifted Rosenbrock's	M	10	−100	100
F57	Shifted rotated Griewank's	M	10	0	600
F58	Shifted rotated Ackley's	M	10	−32	32
F59	Shifted Rastrigin's	M	10	−5	5
F60	Shifted rotated Rastrigin's	M	10	−5	5
F61	Shifted rotated Weierstrass	M	10	−0.5	0.5
F62	Schwefels problem 2.13	M	10	−100	100
F63	Expanded extended Griewank's + Rosenbrock's	E	10	−3	1
F64	Expanded rotated extended Scaffes	E	10	−100	100
F65	Hybrid composition function	HC	10	−5	5
F66	Rotated hybrid comp. Fn 1	HC	10	−5	5
F67	Rotated hybrid comp. Fn 1 with noise	HC	10	−5	5
F68	Rotated hybrid comp. Fn 2	HC	10	−5	5
F69	Rotated hybrid comp. Fn 2 with narrow global optimal	HC	10	−5	5
F70	Rotated hybrid comp. Fn 2 with the global optimum	HC	10	−5	5
F71	Rotated hybrid comp. Fn 3	HC	10	−5	5
F72	Rotated hybrid comp. Fn 3 with high condition number matrix	HC	10	−5	5
F73	Non-continuous rotated hybrid comp. Fn 3	HC	10	−5	5
F74	Rotated hybrid comp. Fn 4	HC	10	−5	5
F75	Rotated hybrid comp. Fn 4	HC	10	−2	5

Table 3
The benchmark problems used in Test 3 (Dim: Dimension, Low, Up: Limits of search space, BC: Bound constrained).

Problem	Name	Type	Low	Up	Dim
F76	Antenna	BC	0.2, 0.2, 0.2, 0.2, 0.2, 0.2, −180, −180, −180, −180, −180, −180,	1, 1, 1, 1, 1, 1, 180, 180, 180, 180, 180, 180	12
F77	Radar	BC	0	$2 \cdot \pi$	20
F78	FM	BC	−6.4	6.35	6

Table 4
Control parameters of the related algorithms used in the tests.

Algorithm	Control parameters			
PSO2011	$C_1 = 1.80$	$C_2 = 1.80$	$\omega = 0.5 + (1 - rand)$	
CMAES	$\sigma = 0.25$	$\mu = \left\lfloor \frac{4 + \lfloor 2 \cdot \log(N) \rfloor}{2} \right\rfloor$		
ABC	$limit = N \cdot D$	Size of employed-bee = (size of colony)/2		
JDE	$F_{inital} = 0.5$	$CR_{initial} = 0.90$	$\tau_1 = 0.1$	$\tau_2 = 0.1$
CLPSO	$c = 1.49445$	$m = 0$	$p_c = 0.5 \cdot \frac{e^t - e^{t(1)}}{e^{t(p_s)} - e^{t(1)}}$ *where*, $t = 0.5 \cdot \left(0 : \frac{1}{p_s - 1} : 1\right)$	
SADE	$F \sim N(0.5, 0.3)$	$CR \sim N(CR_m, 0.1)$	$c = 0.1$	$p = 0.05$
BSA	$mixrate = 1.00$			

solves a specific numerical optimization problem with greater statistical success. This paper uses the global minimum values obtained as the result of 30 runs for its problem-based pairwise comparison of the algorithms. A multi-problem-based pairwise comparison can use the average of the global minimum values obtained as the result of several runs. Multi-problem-based pairwise comparisons determine which algorithm is statistically more successful in a test that includes several benchmark problems [46]. This paper uses the average of global minimum values obtained as the result of 30 runs for its multi-problem-based comparison of the algorithms.

The Wilcoxon Signed-Rank Test was used for pairwise comparisons, with the statistical significance value $\alpha = 0.05$. The null hypothesis H0 for this test is: 'There is no difference between the median of the solutions achieved by algorithm A

Table 5
A simple numerical example describing the functioning of BSA by using F43.

Generation	P (see Eq. (1))		fitnessP	Eq. (3)	oldP (see Eq.s. 2,4)		$F \sim N(0,3)$	Mutant (see Eq. (5))		map		T (see Algorithms 2 and 3)		fitnessT	Globalminimum
Initial	2.713	−4.793	2054.702	0	−3.020	2.605	–	–	–	–	–	–	–	–	Inf
	1.336	2.488	134.179	0	−3.309	−4.117	–	–	–	–	–	–	–	–	
	−0.015	−2.753	199.491	0	1.853	4.533	–	–	–	–	–	–	–	–	
1	2.713	−4.793	2054.702	1	1.336	2.488	−2.473	6.118	−22.799	1	0	2.713	1.741	77.938	77.938
	1.336	2.488	134.179	1	−0.015	−2.753	−2.473	4.677	15.448	0	1	4.677	2.488	2,711.678	
	−0.015	−2.753	199.491	1	2.713	−4.793	−2.473	−6.762	2.291	0	1	−0.582	−2.753	202.178	
2	2.713	1.741	77.938	1	2.713	1.741	0.686	2.713	1.741	1	0	2.713	1.741	77.938	2.500
	1.336	2.488	134.179	1	−0.015	−2.753	0.686	0.409	−1.108	0	1	0.409	2.488	130.140	
	−0.015	−2.753	199.491	1	1.336	2.488	0.686	0.911	0.842	0	0	0.911	0.842	2.005	
3	2.713	1.741	77.938	1	0.911	0.842	7.428	−10.673	−4.937	0	1	−3.489	1.741	357.346	2.500
	0.409	2.488	130.139	1	2.713	1.741	7.428	17.523	−3.061	0	1	−1.159	2.488	128.019	
	0.911	0.842	2.005	1	0.409	2.488	7.428	−2.818	13.068	1	0	0.911	4.442	1,484.491	
4	2.713	1.741	77.938	0	2.713	1.741	4.330	2.713	1.741	0	1	2.713	1.741	77.938	2.005
	−1.159	2.488	128.019	0	0.409	2.488	4.330	5.630	2.488	0	0	1.364	2.488	134.224	
	0.911	0.842	2.005	0	0.911	0.842	4.330	0.911	0.842	0	1	0.911	0.842	2.005	
5	2.713	1.741	77.938	0	0.409	2.488	−0.955	4.913	1.027	1	0	2.713	1.027	51.607	0.307
	−1.159	2.488	128.019	0	0.911	0.842	−0.955	−3.136	4.059	0	1	−3.136	2.488	273.995	
	0.911	0.842	2.005	0	2.713	1.741	−0.955	−0.810	−0.017	0	1	−0.810	0.842	0.307	

Table 6
Basic statistics of the 30-solutions obtained by PSO, CMAES, ABC, JDE, CLPSO, SADE and BSA in Test 1 (Mean: mean-solution, Std: standard-deviation of mean-solution, Best: the best-solution, and Runtime: mean-runtime in seconds.)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
F1	Mean	1.3316029264876300	10.0748846367972000	0.9980038377944500	1.0641405484285200	1.8209961275956800	0.9980038377944500	0.9980038377944500
	Std	0.9455237994690700	8.0277365400340800	0.0000000000000001	0.3622456829347420	1.6979175079427900	0.0000000000000000	0.0000000000000000
	Best	0.9980038377944500	0.9980038377944500	0.9980038377944500	0.9980038377944500	0.9980038377944500	0.9980038377944500	0.9980038377944500
	Runtime	72.527	44.788	64.976	51.101	61.650	66.633	38.125
F2	Mean	2.9999999999999200	21.8999999999995000	3.0000000465423000	2.9999999999999200	3.0000000000000700	2.9999999999999200	2.9999999999999200
	Std	0.0000000000000013	32.6088098948516000	0.0000002350442161	0.0000000000000013	0.0000000000007941	0.0000000000000020	0.0000000000000011
	Best	2.9999999999999200	2.9999999999999200	2.9999999999999200	2.9999999999999200	2.9999999999999200	2.9999999999999200	2.9999999999999200
	Runtime	17.892	24.361	16.624	7.224	24.784	28.699	7.692
F3	Mean	0.1278728062391630	0.0241892995662904	0.0000000000000004	0.0034556340083499	0.0000000000000000	0.0034556340083499	0.0000000000000000
	Std	0.2772792346028400	0.0802240262581864	0.0000000000000001	0.0189272869685522	0.0000000000000000	0.0189272869685522	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	139.555	5.851	84.416	9.492	38.484	15.992	18.922
F4	Mean	0.0043949463343535	0.0003662455278628	0.0000000000000004	0.0007324910557256	0.0000000000000000	0.0440448539086004	0.0000000000000000
	Std	0.0054747064090174	0.0020060093719584	0.0000000000000001	0.0027875840585535	0.0000000000000000	0.2227372747439610	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	126.507	6.158	113.937	14.367	48.667	33.019	24.309
F5	Mean	1.5214322973725000	11.7040011684582000	0.0000000000000340	0.0811017056422860	0.1863456353861950	0.7915368220335460	0.0000000000000105
	Std	0.6617570384662600	9.7201961540865200	0.0000000000000035	0.3176012689149320	0.4389839299322230	0.7561593402959740	0.0000000000000034
	Best	0.0000000000000080	0.0000000000000080	0.0000000000000293	0.0000000000000044	0.0000000000000080	0.0000000000000044	0.0000000000000080
	Runtime	63.039	3.144	23.293	11.016	45.734	40.914	14.396
F6	Mean	0.0000000041922968	0.2540232169641050	0.0000000000000028	0.0000000000000000	0.0000443354499943	0.0000000000000000	0.0000000000000000
	Std	0.0000000139615552	0.3653844307786430	0.0000000000000030	0.0000000000000000	0.0001015919507724	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000005	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	32.409	4.455	22.367	1.279	125.839	4.544	0.962
F7	Mean	0.0000000000000000	0.0622354533647150	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.1345061339146580	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	16.956	6.845	1.832	1.141	2.926	4.409	0.825
F8	Mean	0.0000000000000000	0.0072771062590204	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0398583525142753	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	17.039	2.174	1.804	1.139	2.891	4.417	0.824
F9	Mean	0.0000000000000000	0.0001048363065820	0.0000000000000006	0.0000000000000000	0.0000193464326398	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0005742120996051	0.0000000000000003	0.0000000000000000	0.0000846531630676	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	17.136	2.127	21.713	1.129	33.307	4.303	0.829
F10	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0006005122443674	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0029861918862801	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	17.072	1.375	22.395	1.099	28.508	4.371	0.790
F11	Mean	0.3978873577297380	0.6372170283279430	0.3978873577297380	0.3978873577297380	0.3978873577297390	0.3978873577297380	0.3978873577297380
	Std	0.0000000000000000	0.7302632173480510	0.0000000000000000	0.0000000000000000	0.0000000000000049	0.0000000000000000	0.0000000000000000
	Best	0.3978873577297380	0.3978873577297380	0.3978873577297380	0.3978873577297380	0.3978873577297380	0.3978873577297380	0.3978873577297380
	Runtime	17.049	24.643	10.941	6.814	17.283	27.981	5.450
F12	Mean	0.0000000000000000	0.0000000000000000	0.0715675060725970	0.0000000000000000	0.1593872502094070	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0579425013417103	0.0000000000000000	0.6678482786713720	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0013425253994745	0.0000000000000000	0.0000094069599934	0.0000000000000000	0.0000000000000000

Table 6 (*continued*)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
	Runtime	44.065	1.548	21.487	1.251	166.965	4.405	2.460
F13	Mean	0.6666666666666750	0.6666666666666670	0.0000000000000038	0.6666666666666670	0.0023282133668190	0.6666666666666670	0.6444444444444440
	Std	0.0000000000000022	0.0000000000000000	0.0000000000000012	0.0000000000000002	0.0051792840882291	0.0000000000000000	0.1217161238900370
	Best	0.6666666666666720	0.6666666666666670	0.0000000000000021	0.6666666666666670	0.0000120708732167	0.6666666666666670	0.0000000000000000
	Runtime	167.094	3.719	37.604	18.689	216.261	47.833	21.192
F14	Mean	−1.0000000000000000	−0.1000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000
	Std	0.0000000000000000	0.3051285766293650	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000
	Runtime	16.633	3.606	13.629	6.918	16.910	28.739	5.451
F15	Mean	0.0000000000000000	1028.3930784026900000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	1298.1521820113500000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	27.859	15.541	40.030	2.852	4.030	6.020	2.067
F16	Mean	48.7465164446927000	1680.3460230073400000	0.0218688498331872	0.9443728655432830	81.7751618148164000	0.0000000000000000	0.0000000000000000
	Std	88.8658510972991000	2447.7484859066000000	0.0418409568792831	2.8815514827061600	379.9241117377270000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000016	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	95.352	11.947	44.572	4.719	162.941	5.763	7.781
F17	Mean	918.9518492782850000	12340.2283326398000000	11.0681496253548000	713.7226974626920000	0.8530843976878610	0.0000000000000000	0.0000000000000000
	Std	1652.4810858411400000	22367.1698875802000000	9.8810950146557100	1710.0713074301200000	2.9208253191698800	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.3274654777056860	0.0000000000000000	0.0016957837829822	0.0000000000000000	0.0000000000000000
	Runtime	271.222	7.631	43.329	16.105	268.894	168.310	33.044
F18	Mean	0.0068943694819713	0.0011498935321349	0.0000000000000000	0.0048193578543185	0.0000000000000000	0.0226359326967139	0.0004930693556077
	Std	0.0080565201649587	0.0036449413521107	0.0000000000000001	0.0133238235582874	0.0000000000000000	0.0283874287215679	0.0018764355751644
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	73.895	2.647	19.073	6.914	14.864	25.858	5.753
F19	Mean	−3.8627821478207500	−3.7243887744664700	−3.8627821478207500	−3.8627821478207500	−3.8627821478207500	−3.8627821478207500	−3.8627821478207500
	Std	0.0000000000000027	0.5407823545193820	0.0000000000000024	0.0000000000000027	0.0000000000000027	0.0000000000000027	0.0000000000000027
	Best	−3.8627821478207600	−3.8627821478207600	−3.8627821478207600	−3.8627821478207600	−3.8627821478207600	−3.8627821478207600	−3.8627821478207600
	Runtime	19.280	21.881	12.613	7.509	17.504	24.804	6.009
F20	Mean	−3.3180320675402500	−3.2942534432762600	−3.3219951715842400	−3.2982165473202600	−3.3219951715842400	−3.3140689634962500	−3.3219951715842400
	Std	0.0217068148263721	0.0511458075926848	0.0000000000000014	0.0483702518391572	0.0000000000000013	0.0301641516823498	0.0000000000000013
	Best	−3.3219951715842400	−3.3219951715842400	−3.3219951715842400	−3.3219951715842400	−3.3219951715842400	−3.3219951715842400	−3.3219951715842400
	Runtime	26.209	7.333	13.562	8.008	20.099	33.719	6.822
F21	Mean	0.0003074859878056	0.0064830287538208	0.0004414866359626	0.0003685318137604	0.0003100479704151	0.0003074859878056	0.0003074859878056
	Std	0.0000000000000000	0.0148565973286009	0.0000568392289725	0.0002323173367683	0.0000059843325073	0.0000000000000000	0.0000000000000000
	Best	0.0003074859878056	0.0003074859878056	0.0003230956007045	0.0003074859878056	0.0003074859941292	0.0003074859878056	0.0003074859878056
	Runtime	84.471	13.864	20.255	7.806	156.095	45.443	11.722
F22	Mean	−1.0809384421344400	−0.7323679641701760	−1.0809384421344400	−1.0764280762657400	−1.0202940450426400	−1.0809384421344400	−1.0809384421344400
	Std	0.0000000000000006	0.4136688304155380	0.0000000000000008	0.0247042912888477	0.1190811583120530	0.0000000000000005	0.0000000000000005
	Best	−1.0809384421344400	−1.0809384421344400	−1.0809384421344400	−1.0809384421344400	−1.0809384421344400	−1.0809384421344400	−1.0809384421344400
	Runtime	27.372	32.311	27.546	19.673	52.853	36.659	21.421
F23	Mean	−1.3891992200744600	−0.5235864386288060	−1.4999990070800800	−1.3431399432579700	−1.4765972735526500	−1.4999992233525000	−1.4821658762555300
	Std	0.2257194403158630	0.2585330714077300	0.0000008440502079	0.2680292304904580	0.1281777579497830	0.0000000000000009	0.0976772648082733
	Best	−1.4999992233524900	−0.7977041047646610	−1.4999992233524900	−1.4999992233524900	−1.4999992233524900	−1.4999992233524900	−1.4999992233524900
	Runtime	33.809	17.940	37.986	20.333	42.488	36.037	18.930
F24	Mean	−0.9166206788680230	−0.3105071678265780	−0.8406348096500680	−0.8827152798835760	−0.9431432797743700	−1.2765515661973800	−1.3127183561646500
	Std	0.3917752367440500	0.2080317241440800	0.2000966365984320	0.3882445165494030	0.3184175870987750	0.3599594108130040	0.3158807699946290
	Best	−1.5000000000003800	−0.7976938356122860	−1.4999926800631400	−1.5000000000003800	−1.5000000000003800	−1.5000000000003800	−1.5000000000003800

(*continued on next page*)

Table 6 (*continued*)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
	Runtime	110.798	8.835	38.470	21.599	124.609	47.171	35.358
F25	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000004	0.0000000000000000	0.0000041787372626	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000161643637543	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	25.358	1.340	19.689	1.142	31.632	4.090	0.813
F26	Mean	−1.8210436836776800	−1.7829268228561700	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800
	Std	0.0000000000000009	0.1450583631808370	0.0000000000000009	0.0000000000000009	0.0000000000000009	0.0000000000000009	0.0000000000000009
	Best	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800	−1.8210436836776800
	Runtime	19.154	26.249	17.228	9.663	18.091	28.453	7.472
F27	Mean	−4.6565646397053900	−4.1008953007033700	−4.6934684519571100	−4.6893456932617100	−4.6920941990586400	−4.6884965299983800	−4.6934684519571100
	Std	0.0557021530063238	0.4951250481844850	0.0000000000000009	0.0125797149251589	0.0075270931220834	0.0272323381095561	0.0000000000000008
	Best	−4.6934684519571100	−4.6934684519571100	−4.6934684519571100	−4.6934684519571100	−4.6934684519571100	−4.6934684519571100	−4.6934684519571100
	Runtime	38.651	10.956	17.663	14.915	25.843	38.446	11.971
F28	Mean	−8.9717330307549300	−7.6193507368464700	−9.6601517156413500	−9.6397230986132500	−9.6400278592589600	−9.6572038232921700	−9.6601517156413500
	Std	0.4927013165009220	0.7904830398850970	0.0000000000000008	0.0393668145094111	0.0437935551332868	0.0105890022905617	0.0000000000000007
	Best	−9.5777818097208200	−9.1383975057875100	−9.6601517156413500	−9.6601517156413500	−9.6601517156413500	−9.6601517156413500	−9.6601517156413500
	Runtime	144.093	6.959	27.051	20.803	32.801	46.395	22.250
F29	Mean	0.0119687224560441	0.0788734736114700	0.0838440014038032	0.0154105130055856	0.0198686590210374	0.0140272066690658	0.0007283694780796
	Std	0.0385628598040034	0.1426911799629180	0.0778327303965192	0.0308963906374663	0.0613698943155661	0.0328868042987376	0.0014793717464195
	Best	0.0000044608370213	0.0000000000000000	0.0129834451730589	0.0000000000000000	0.0000175219764526	0.0000000000000000	0.0000000000000000
	Runtime	359.039	17.056	60.216	35.044	316.817	92.412	191.881
F30	Mean	0.0000130718912008	0.0000000000000000	0.0002604330013462	0.0000000000000001	0.0458769685199585	0.0000002733806735	0.0000000028443186
	Std	0.0000014288348929	0.0000000000000000	0.0000394921919294	0.0000000000000002	0.0620254411839524	0.0000001788830279	0.0000000033308990
	Best	0.0000095067504097	0.0000000000000000	0.0001682411286088	0.0000000000000000	0.0005277712020642	0.0000000944121661	0.0000000004769768
	Runtime	567.704	14.535	215.722	194.117	252.779	360.380	144.784
F31	Mean	0.0001254882834238	0.0000000000000000	0.0077905311094958	0.0020185116261490	0.0002674563703837	0.0000000000000000	0.0000000111676630
	Std	0.0001503556280087	0.0000000000000000	0.0062425841086448	0.0077448684015362	0.0003044909265796	0.0000000000000000	0.0000000184322163
	Best	0.0000000156460198	0.0000000000000000	0.0003958766023752	0.0000000000000000	0.0000023064754605	0.0000000000000000	0.0000000000000000
	Runtime	250.248	12.062	34.665	48.692	227.817	220.886	149.882
F32	Mean	0.0003548345513179	0.0701619169853449	0.0250163252527030	0.0013010316180679	0.0019635752485802	0.0016730768406953	0.0019955316015528
	Std	0.0001410817500914	0.0288760292572957	0.0077209314806873	0.0009952078711752	0.0043423828633839	0.0007330246909835	0.0009698942217908
	Best	0.0001014332605364	0.0299180701536354	0.0094647580732654	0.0001787238105452	0.0004206447422138	0.0005630852254632	0.0006084880639553
	Runtime	290.669	2.154	34.982	82.124	103.283	171.637	48.237
F33	Mean	25.6367602258676000	95.9799861204982000	0.0000000000000000	1.1276202647057400	0.6301407361590880	0.8622978494808570	0.0000000000000000
	Std	8.2943512684216700	56.6919245985100000	0.0000000000000000	1.0688393637536800	0.8046401822326410	0.9323785263847000	0.0000000000000000
	Best	12.9344677422129000	29.8487565993415000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	76.083	2.740	4.090	7.635	18.429	23.594	5.401
F34	Mean	2.6757043114269700	0.3986623855035210	0.2856833465904130	1.0630996944802500	5.7631786582751800	1.2137377447007000	0.3986623854300930
	Std	12.3490058210004000	1.2164328621946200	0.6247370987465170	1.7930895051734300	13.9484817304201000	1.8518519388285700	1.2164328622195200
	Best	0.0042535368984501	0.0000000000000000	0.0004266049929880	0.0000000000000000	0.0268003205820685	0.0001448955835246	0.0000000000000000
	Runtime	559.966	9.462	35.865	23.278	187.894	268.449	34.681
F35	Mean	0.0000000000000000	0.4651202457398910	0.0000000000000000	0.0038863639514140	0.0019431819755029	0.0006477273251676	0.0000000000000000
	Std	0.0000000000000000	0.0933685176073728	0.0000000000000000	0.0048411743884718	0.0039528023354469	0.0024650053428137	0.0000000000000000
	Best	0.0000000000000000	0.0097159098775144	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	18.163	24.021	7.861	4.216	8.304	5.902	1.779
F36	Mean	−7684.6104757783800000	−6835.1836730901400000	−12569.4866181730000000	−12304.9743375341000000	−12210.8815698372000000	−12549.7468957373000000	−12569.4866181730000000
	Std	745.3954005014180000	750.7338055436110000	0.0000000000022659	221.4322514436480000	205.9313376284770000	44.8939348779747000	0.0000000000024122
	Best	−8912.8855854978200000	−8340.0386911070600000	−12569.4866181730000000	−12569.4866181730000000	−12569.4866181730000000	−12569.4866181730000000	−12569.4866181730000000

Table 6 (continued)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
	Runtime	307.427	3.174	19.225	10.315	31.499	34.383	11.069
F37	Mean	0.0000000000000000	0.0000000000000000	14.5668734126948000	0.0000000000000000	6.4655746330439100	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	8.7128443012950300	0.0000000000000000	8.2188901353055800	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	4.0427699323673400	0.0000000000000000	0.1816624029553790	0.0000000000000000	0.0000000000000000
	Runtime	543.180	3.370	111.841	19.307	179.083	109.551	57.294
F38	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000005	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	163.188	2.558	20.588	1.494	12.563	5.627	3.208
F39	Mean	−10.1061873621653000	−5.2607563471326400	−10.5364098166920000	−10.3130437162426000	−10.3130437162026000	−10.5364098166921000	−10.5364098166921000
	Std	1.6679113661236400	3.6145751818694000	0.0000000000000023	1.2234265179812200	1.2234265179736500	0.0000000000000016	0.0000000000000018
	Best	−10.5364098166921000	−10.5364098166921000	−10.5364098166920000	−10.5364098166921000	−10.5364098166920000	−10.5364098166921000	−10.5364098166920000
	Runtime	31.018	11.024	16.015	8.345	37.275	28.031	7.045
F40	Mean	−9.5373938082045500	−5.7308569926624600	−10.1531996790582000	−9.5656135761215700	−10.1531996790582000	−9.9847854277673500	−10.1531996790582000
	Std	1.9062127067994200	3.5141202468383400	0.0000000000000055	1.8315977756329900	0.0000000000000076	0.9224428443735560	0.0000000000000072
	Best	−10.1531996790582000	−10.1531996790582000	−10.1531996790582000	−10.1531996790582000	−10.1531996790582000	−10.1531996790582000	−10.1531996790582000
	Runtime	25.237	11.177	11.958	7.947	30.885	25.569	6.864
F41	Mean	−10.4029405668187000	−6.8674070870953700	−10.4029405668187000	−9.1615813354737300	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000
	Std	0.0000000000000018	3.6437803702691000	0.0000000000000006	2.8277336448396200	0.0000000000000010	0.0000000000000018	0.0000000000000017
	Best	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000	−10.4029405668187000
	Runtime	21.237	11.482	14.911	8.547	31.207	27.064	8.208
F42	Mean	−186.7309073569880000	−81.5609772893002000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000
	Std	0.0000046401472660	66.4508342743478000	0.0000000000000236	0.0000000000000388	0.0000000000000279	0.0000000000000377	0.0000000000000224
	Best	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000	−186.7309088310240000
	Runtime	19.770	25.225	13.342	8.213	20.344	27.109	9.002
F43	Mean	−1.0316284534898800	−1.0044229658530100	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800
	Std	0.0000000000000005	0.1490105926664260	0.0000000000000005	0.0000000000000005	0.0000000000000005	0.0000000000000005	0.0000000000000005
	Best	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800	−1.0316284534898800
	Runtime	16.754	24.798	11.309	7.147	18.564	27.650	5.691
F44	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000004	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	159.904	2.321	21.924	1.424	14.389	5.920	3.302
F45	Mean	2.3000000000000000	0.0666666666666667	0.0000000000000000	0.9000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	1.8597367258983700	0.2537081317024630	0.0000000000000000	3.0211895350832500	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	57.276	1.477	1.782	2.919	3.042	4.307	0.883
F46	Mean	0.1333333333333330	0.2666666666666670	0.0000000000000000	0.0000000000000000	0.2000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.3457459036417600	0.9444331755018490	0.0000000000000000	0.0000000000000000	0.4068381021724860	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	20.381	2.442	1.700	1.074	6.142	4.319	0.764
F47	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000005	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000003	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	564.178	2.565	24.172	1.870	15.948	6.383	4.309
F48	Mean	−50.0000000000002000	−50.0000000000002000	−49.9999999999997000	−50.0000000000002000	−49.4789234062579000	−50.0000000000002000	−50.0000000000002000
	Std	0.0000000000000361	0.0000000000000268	0.0000000000001408	0.0000000000000354	1.3150773145311700	0.0000000000000268	0.0000000000000361
	Best	−50.0000000000002000	−50.0000000000002000	−50.0000000000001000	−50.0000000000002000	−49.9999994167392000	−50.0000000000002000	−50.0000000000002000

(continued on next page)

Table 6 (*continued*)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
	Runtime	24.627	8.337	22.480	8.623	142.106	36.804	7.747
F49	Mean	−210.0000000000010000	−210.0000000000030000	−209.9999999999470000	−210.0000000000030000	−199.5925885475030000	−210.0000000000030000	−210.0000000000030000
	Std	0.0000000000009434	0.0000000000003702	0.0000000000138503	0.0000000000008251	9.6415263953591700	0.0000000000004625	0.0000000000003950
	Best	−210.0000000000030000	−210.0000000000030000	−209.9999999999690000	−210.0000000000040000	−209.9858674090290000	−210.0000000000040000	−210.0000000000040000
	Runtime	48.580	5.988	36.639	11.319	187.787	54.421	11.158
F50	Mean	0.0000000000000000	0.0000000000000000	0.0000000402380424	0.0000000000000000	0.0000000001597805	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000002203520334	0.0000000000000000	0.0000000006266641	0.0000000000000000	0.0000000000000000
	Best	0.0000000000000000	0.0000000000000000	0.0000000000000210	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	86.369	1.868	86.449	1.412	157.838	4.930	5.702

Table 7
Determining the algorithm that statistically provides the best solution for each benchmark problem used in Test 1 by utilizing two-sided Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Problem	PSO vs. BSA				CMAES vs. BSA				ABC vs. BSA				JDE vs. BSA				CLPSO vs. BSA				SADE vs. BSA			
	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner
F1	1.56E−02	0	28	+	8.24E−06	0	351	+	1.56E−02	0	28	+	1.00E+00	0	1	=	1.95E−03	0	55	+	1.00E+00	0	0	=
F2	5.73E−07	0	325	+	1.28E−06	0	435	+	4.19E−06	12	366	+	8.77E−05	40.5	310.5	+	1.71E−04	34.5	290.5	+	2.39E−04	37.5	262.5	+
F3	2.44E−04	0	91	+	1.25E−01	0	10	=	8.87E−07	0	465	+	1.00E+00	0	1	=	1.00E+00	0	0	=	1.00E+00	0	1	=
F4	4.88E−04	0	78	+	1.00E+00	0	1	=	9.92E−07	0	465	+	5.00E−01	0	3	=	1.00E+00	0	0	=	1.56E−02	0	28	+
F5	2.97E−06	1.5	433.5	+	2.25E−04	21	279	+	1.23E−06	0	465	+	2.44E−04	49.5	55.5	+	9.15E−01	66	70	=	3.51E−03	91	374	+
F6	3.79E−06	0	406	+	1.95E−03	0	55	+	1.71E−06	0	465	+	1.00E+00	0	0	=	2.70E−05	0	276	+	1.00E+00	0	0	=
F7	1.00E+00	0	0	=	1.56E−02	0	28	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F8	1.00E+00	0	0	=	1.00E+00	0	1	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F9	1.00E+00	0	0	=	1.00E+00	0	1	=	2.43E−06	0	435	+	1.00E+00	0	0	=	2.70E−05	0	276	+	1.00E+00	0	0	=
F10	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	5.61E−06	0	378	+	1.00E+00	0	0	=
F11	1.00E+00	0	0	=	2.50E−01	0	6	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	1	=	1.00E+00	0	0	=
F12	1.00E+00	0	0	=	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=
F13	1.34E−06	0	465	+	1.00E+00	0	1	=	7.84E−07	464	1	–	7.81E−03	0	36	+	1.92E−06	464	1	–	1.00E+00	0	1	=
F14	1.00E+00	0	0	=	2.03E−07	0	378	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F15	1.00E+00	0	0	=	6.10E−05	0	120	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F16	4.88E−04	0	78	+	8.20E−06	0	351	+	1.73E−06	0	465	+	2.50E−01	0	6	=	4.88E−04	0	78	+	1.00E+00	0	0	=
F17	5.96E−05	0	231	+	3.79E−06	0	406	+	1.73E−06	0	465	+	6.10E−05	0	120	+	1.73E−06	0	465	+	1.00E+00	0	0	=
F18	3.24E−05	12	339	+	2.50E−01	2	8	=	2.44E−04	27	78	+	9.77E−04	12.5	65.5	+	5.00E−01	3	0	=	1.40E−04	3	207	+
F19	1.00E+00	0	0	=	7.81E−03	0	36	+	1.62E−06	0	276	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F20	1.00E+00	0	1	=	1.56E−02	0	28	+	1.00E+00	0	0	=	3.13E−02	0	21	+	1.00E+00	0	0	=	5.00E−01	0	3	=
F21	1.00E+00	0	0	=	3.58E−05	0	253	+	1.73E−06	0	465	+	5.00E−01	0	3	=	1.73E−06	0	465	+	1.00E+00	0	0	=
F22	1.00E+00	0	0	=	4.32E−04	0	136	+	9.77E−04	0	66	+	1.00E+00	0	1	=	3.78E−06	0	406	+	1.00E+00	0	0	=
F23	6.25E−02	0	15	=	1.73E−06	0	465	+	3.61E−05	29	406	+	7.81E−03	2.5	42.5	+	1.00E+00	1	2	=	1.00E+00	1	0	=
F24	2.91E−04	15	238	+	1.92E−06	1	464	+	2.16E−05	26	439	+	1.61E−04	18	282	+	2.69E−04	49	386	+	1.22E−04	58	62	+
F25	1.00E+00	0	0	=	1.00E+00	0	0	=	3.56E−06	0	406	+	1.00E+00	0	0	=	1.23E−05	0	325	+	1.00E+00	0	0	=
F26	1.00E+00	0	0	=	1.25E−01	0	10	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F27	7.06E−05	0	190	+	2.49E−06	0	435	+	1.00E+00	0	0	=	2.50E−01	0	6	=	1.00E+00	0	1	=	1.00E+00	0	1	=
F28	1.73E−06	0	465	+	1.73E−06	0	465	+	1.00E+00	0	0	=	2.44E−04	0	91	+	9.77E−04	0	66	+	2.50E−01	0	6	=
F29	6.42E−03	100	365	+	1.87E−04	33.5	344.5	+	1.73E−06	0	465	+	1.57E−04	37	369	+	4.53E−04	62	403	+	3.76E−04	41	337	+
F30	1.73E−06	0	465	+	1.73E−06	465	0	–	1.73E−06	0	465	+	1.73E−06	465	0	–	1.73E−06	0	465	+	1.73E−06	0	465	+
F31	1.73E−06	0	465	+	2.56E−06	435	0	-	1.73E−06	0	465	+	5.19E−04	378	57	–	1.73E−06	0	465	+	2.56E−06	435	0	–
F32	1.73E−06	465	0	–	1.73E−06	0	465	+	1.73E−06	0	465	+	1.48E−02	351	114	–	9.27E−03	359	106	–	2.29E−01	291	174	=
F33	1.73E−06	0	465	+	1.73E−06	0	465	+	1.00E+00	0	0	=	3.56E−05	0	231	+	1.22E−04	0	105	+	2.00E−04	0	153	+
F34	2.22E−04	53	412	+	6.25E−02	9	12	=	2.77E−03	87	378	+	3.91E−03	8	47	+	1.13E−05	19	446	+	3.88E−04	60	405	+
F35	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=	4.88E−04	0	78	+	3.13E−02	0	21	+	5.00E−01	0	3	=
F36	1.73E−06	0	465	+	1.73E−06	0	465	+	1.00E+00	0	0	=	1.08E−05	0	325	+	4.75E−06	0	378	+	6.25E−02	0	15	=
F37	1.00E+00	0	0	=	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=
F38	1.00E+00	0	0	=	1.00E+00	0	0	=	8.31E−07	0	465	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F39	5.00E−01	0	3	=	5.45E−05	0	231	+	1.56E−02	0	28	+	1.00E+00	0	1	=	2.50E−01	0	6	=	1.00E+00	0	0	=
F40	2.50E−01	0	6	=	1.02E−04	0	190	+	1.00E+00	0	0	=	2.50E−01	0	6	=	5.00E−01	0	3	=	1.00E+00	0	1	=
F41	1.00E+00	0	0	=	6.10E−05	0	120	+	5.00E−01	0	3	=	6.25E−02	0	15	=	5.00E−01	0	3	=	1.00E+00	0	0	=
F42	2.41E−06	0	435	+	1.71E−06	0	465	+	1.95E−03	0	55	+	2.44E−04	0	91	+	1.31E−05	0	190	+	4.65E−04	8.5	127.5	+
F43	1.00E+00	0	0	=	1.00E+00	0	1	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F44	1.00E+00	0	0	=	1.00E+00	0	0	=	1.23E−06	0	465	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F45	2.93E−06	0	406	+	5.00E−01	0	3	=	1.00E+00	0	0	=	1.56E−02	0	28	+	1.00E+00	0	0	=	1.00E+00	0	0	=
F46	1.25E−01	0	10	=	1.25E−01	0	10	=	1.00E+00	0	0	=	1.00E+00	0	0	=	3.13E−02	0	21	+	1.00E+00	0	0	=
F47	1.00E+00	0	0	=	1.00E+00	0	0	=	8.12E−07	0	465	+	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=

(*continued on next page*)

Table 7 (*continued*)

Problem	PSO vs. BSA				CMAES vs. BSA				ABC vs. BSA				JDE vs. BSA				CLPSO vs. BSA				SADE vs. BSA			
	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner
F48	1.00E+00	0	0	=	7.81E−03	0	36	+	1.65E−06	0	465	+	5.00E−01	3	0	=	1.73E−06	0	465	+	2.73E−06	253	0	−
F49	4.38E−05	14	311	+	3.91E−03	22	33	+	1.73E−06	0	465	+	6.07E−01	96.5	74.5	=	1.73E−06	0	465	+	2.88E−04	163	8	−
F50	1.00E+00	0	0	=	1.00E+00	0	0	=	1.73E−06	0	465	+	1.00E+00	0	0	=	4.38E−04	0	136	+	1.00E+00	0	0	=
+/=/−	22/27/1				28/20/2				31/18/1				16/31/3				26/22/2				10/37/3			

Table 8
Basic statistics of the 30-solutions obtained by PSO, CMAES, ABC, JDE, CLPSO, SADE and BSA in Test 2 (Mean: mean-solution, Std: standard-deviation of mean-solution, Best: the best-solution, and Runtime: mean-runtime in seconds.)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
F51	Mean	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Best	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000
	Runtime	212.862	23.146	113.623	118.477	167.675	154.232	140.736
F52	Mean	−450.0000000000000000	−450.0000000000000000	−449.9999999999220000	−450.0000000000000000	−418.8551838547760000	−450.0000000000000000	−450.0000000000000000
	Std	0.0000000000000350	0.0000000000000000	0.0000000002052730	0.0000000000000615	51.0880511039985000	0.0000000000000000	0.0000000000000259
	Best	−450.0000000000000000	−450.0000000000000000	−449.9999999999970000	−450.0000000000000000	−449.4789299923810000	−450.0000000000000000	−450.0000000000000000
	Runtime	230.003	23.385	648.784	139.144	1462.706	185.965	243.657
F53	Mean	−44.5873911956554000	−450.0000000000000000	387131.2441213970000000	−197.9999999999850000	62142.8213760465000000	245.0483283713550000	−449.9999567867430000
	Std	458.5794120016290000	0.0000000000000000	166951.7336592640000000	391.5169437474990000	34796.1785167236000000	790.6056596723160000	0.0001175386756044
	Best	−443.9511286079800000	−450.0000000000000000	165173.1853095600000000	−449.9999999999990000	17306.9066792474000000	−421.4054944641620000	−450.0000000000000000
	Runtime	2658.937	35.464	240.094	1017.557	1789.643	1808.954	1883.713
F54	Mean	−450.0000000000000000	77982.4567046980000000	140.4509447125110000	−414.0000000000000000	−178.8320689185280000	−450.0000000000000000	−450.0000000000000000
	Std	0.0000000000000460	131376.7365456010000000	217.2646715063190000	55.9309919639279000	394.8667499339530000	0.0000000000000000	0.0000000000000259
	Best	−450.0000000000000000	−450.0000000000000000	−324.3395691109350000	−450.0000000000000000	−447.9901256558030000	−450.0000000000000000	−450.0000000000000000
	Runtime	247.256	32.726	209.188	143.767	1248.616	185.438	347.167
F55	Mean	−310.0000000000000000	−310.0000000000000000	−291.5327549384120000	−271.0000000000000000	333.4108259915760000	−309.9999999999960000	−309.9999999999980000
	Std	0.0000000000000000	0.0000000000000000	17.6942171217937000	60.5919079609218000	512.6920837704510000	0.0000000000133965	0.0000000000023443
	Best	−310.0000000000000000	−310.0000000000000000	−307.7611364354020000	−310.0000000000000000	−309.9740055344430000	−310.0000000000000000	−310.0000000000000000
	Runtime	241.517	39.293	205.568	134.078	1481.686	210.684	386.633
F56	Mean	393.4959999056240000	390.5315438816460000	391.2531452421960000	231.3986579112350000	405.5233436479650000	390.2657719408230000	390.1328859704120000
	Std	16.0224965900462000	1.3783433976378300	3.7254660805238600	247.2968415284400000	10.7480096852869000	1.0114275384776600	0.7278464357038200
	Best	390.0000000000150000	390.0000000000000000	390.0101471658490000	−140.0000000000000000	390.5776683413440000	390.0000000000000000	390.0000000000000000
	Runtime	1178.079	27.894	159.762	153.715	1441.859	1214.303	290.236
F57	Mean	1091.0644335162500000	1087.2645466786700000	1087.0459486286000000	1141.0459486286000000	1087.0459486286000000	1087.0459486286000000	1087.0459486286000000
	Std	3.4976948942723200	0.5365230018001780	0.0000000000005585	83.8964879458918000	0.0000000000004264	0.0000000000004814	0.0000000000004428
	Best	1087.0696772583000000	1087.0459486286000000	1087.0459486286000000	1087.0459486286000000	1087.0459486286000000	1087.0459486286000000	1087.0459486286000000
	Runtime	334.064	37.047	180.472	159.922	267.342	259.760	332.132
F58	Mean	−119.8190232990920000	−119.9261073509850000	−119.7446063439080000	−119.4450938018030000	−119.9300269839980000	−119.7727713703720000	−119.8356122057440000
	Std	0.0720107560874199	0.1554021446157740	0.0623866434489108	0.0927418223065644	0.0417913553101429	0.1248514853682450	0.0704515460477787
	Best	−119.9302772694110000	−120.0000000000000000	−119.8779554779730000	−119.6575717927190000	−119.9756745390830000	−119.9999999999980000	−119.9802847896350000
	Runtime	602.507	49.209	265.319	160.806	1586.286	648.489	717.375
F59	Mean	−324.6046006320200000	−306.5782069681560000	−330.0000000000000000	−329.8673387923880000	−329.4361898676470000	−329.9668346980970000	−330.0000000000000000
	Std	2.5082306041521000	21.9475396048756000	0.0000000000000000	0.3440030182812760	0.6229063711904190	0.1816538397880230	0.0000000000000000
	Best	−329.0050409429070000	−327.0151228287200000	−330.0000000000000000	−330.0000000000000000	−330.0000000000000000	−330.0000000000000000	−330.0000000000000000
	Runtime	982.449	22.237	111.629	128.494	162.873	155.645	176.994
F60	Mean	−324.3311322538170000	−314.7871102989330000	−306.7949047862760000	−319.6763749798700000	−321.7278926895280000	−322.9689591871600000	−319.2544515903510000
	Std	3.0072222933667300	8.3115989308305500	5.1787864195870400	4.9173541245304800	1.8971778613701300	2.8254645254663600	3.3091959975390800
	Best	−327.1650513120000000	−327.0151228287200000	−318.9403196374510000	−326.0201637716270000	−326.1788303102740000	−328.0100818858130000	−325.0252097523530000
	Runtime	1146.013	29.860	259.258	179.039	1594.096	210.534	420.851
F61	Mean	92.5640111212146000	90.7642785704506000	94.8428485804138000	93.2972315784963000	94.6109567642977000	91.6859083842723000	92.3519494286347000
	Std	1.5827416781636900	26.4613831425879000	0.6869412813090850	1.8766951726453600	0.6689129174038950	0.9033073777915270	1.0901581870340800
	Best	90.1142082473923000	−45.0054133586912000	93.1500794016147000	91.0295373630387000	92.9690673344598000	90.1363685040678000	90.2628852415150000
	Runtime	1310.457	44.217	308.501	282.150	1421.545	506.829	1771.860
F62	Mean	18611.3142254809000000	−70.0486708747625000	−337.3273080760500000	400.3240208136310000	−447.8870804905020000	−394.5206365378250000	−437.1125728026770000
	Std	12508.7866126316000000	637.4585182420270000	56.5730759032367000	688.3344299264300000	11.8934815947019000	128.6353424718180000	20.3541618366546000
	Best	4568.3350537809200000	−460.0000000000000000	−449.1707421778360000	−434.8788220982740000	−459.6890294276810000	−460.0000000000000000	−459.1772521346520000
	Runtime	2381.974	34.857	232.916	202.941	1636.440	1277.975	1466.985

(*continued on next page*)

Table 8 (*continued*)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
F63	Mean	−129.2373581503910000	−128.7850616923410000	−129.8343428775830000	−129.6294851450880000	−129.8382867796110000	−129.7129164862680000	−129.8981409848090000
	Std	0.5986210944493790	0.6157633658946230	0.0408016481905455	0.1054759371085400	0.0372256921835666	0.0875456568200232	0.0682328484314248
	Best	−129.6861385930680000	−129.5105509483130000	−129.9098920058450000	−129.8125711770830000	−129.9098505660780000	−129.8717592632560000	−129.9901230990300000
	Runtime	2183.218	25.496	205.194	186.347	1526.365	660.986	1064.114
F64	Mean	−298.2835926212850000	−295.1290938304830000	−296.9323391084610000	−296.8839733969750000	−297.5119726691150000	−297.8403738182600000	−297.5359077431460000
	Std	0.5587676271753680	0.1634039984609270	0.2251930667702880	0.4330673614598290	0.3440115280624180	0.4536801689800720	0.4085859316264990
	Best	−299.6022022972560000	−295.7382222729600000	−297.4659619544820000	−297.8411886637500000	−298.3030560759620000	−299.2417795907860000	−298.3869295150680000
	Runtime	2517.138	32.084	262.533	334.888	1615.452	1289.814	1953.289
F65	Mean	417.4613663019860000	492.5045364088000000	120.0000000000000000	326.6601114362900000	131.3550392249760000	234.2689845349590000	120.0000000000000000
	Std	153.9215808771580000	181.5709657779580000	0.0000000000000188	174.6877238188330000	26.1407360548431000	150.7595974059750000	0.0000000000000000
	Best	120.0000000000000000	262.7619554120320000	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000
	Runtime	3156.336	239.823	2285.787	1834.967	3210.655	1932.016	2351.478
F66	Mean	221.4232628350220000	455.1151684594550000	258.8582688922670000	231.1806131539990000	231.5547154800990000	222.0256674919140000	234.4843380488580000
	Std	12.2450207482898000	254.3583511786970000	11.8823213189685000	13.5473380962764000	11.5441451076421000	6.1841489800660300	8.9091119100451100
	Best	181.5746616282570000	120.0000000000000000	235.6600739998890000	210.3582705649860000	214.7661703584830000	206.4520786020840000	219.6244910167680000
	Runtime	4242.280	202.808	2237.308	1824.388	8649.998	2970.950	8270.920
F67	Mean	217.3338617866620000	681.0349114021570000	265.0370119084380000	228.7309024901770000	240.3635189964930000	221.1801916743850000	228.3769828342800000
	Std	20.6685850658838000	488.0618274343640000	12.4033917090208000	12.3682716268631000	14.8435137485293000	5.7037006844690500	8.7086794471239900
	Best	120.0000000000000000	223.0782617790520000	241.9810089596350000	181.6799927773160000	221.3817133141830000	209.2509748304710000	204.6479138174220000
	Runtime	8208.697	197.497	2159.392	5873.112	4599.027	5938.879	8189.243
F68	Mean	668.9850326105730000	926.9488078829420000	513.8925774904480000	743.9859973770210000	892.4391527217660000	845.4504613493740000	587.5732354221340000
	Std	275.8071370273340000	174.1027182659660000	31.0124861524005000	175.6497294240330000	79.1422224454971000	120.8505129523180000	250.0556329707140000
	Best	310.0000000000000000	310.0000000000000000	444.4692044973030000	310.0000000000000000	738.3764781625320000	310.0000000000000000	310.0000000000000000
	Runtime	3687.235	251.155	2445.259	1777.638	8398.690	3073.274	4554.102
F68	Mean	708.2979222913040000	831.2324139697050000	500.5478931040730000	776.5150806087790000	863.8926908090610000	809.7183195902260000	587.6511686191670000
	Std	256.2419561521300000	250.1848775931620000	31.2240894705539000	160.7307526692470000	96.5618989087194000	147.3158109824600000	236.1141037692630000
	Best	310.0000000000000000	310.0000000000000000	407.3155842366960000	363.8314566805740000	493.0042540796450000	310.0000000000000000	310.0000000000000000
	Runtime	5258.509	222.015	2341.791	1849.670	9909.479	3213.601	4764.968
F70	Mean	711.2970397614200000	876.9306188768990000	483.2984167460740000	761.2954767038960000	844.6391674419360000	810.5227124472170000	612.0906184834040000
	Std	258.9317052508320000	289.7296413284470000	99.3976740616107000	163.4084080635650000	113.6848457105400000	104.7139423525340000	249.5599278421970000
	Best	310.0000000000000000	310.0000000000000000	155.5049931377980000	363.8314568648180000	489.0742585970560000	310.0000000000000000	310.0000000000000000
	Runtime	4346.055	228.619	2250.917	1900.279	9988.261	2818.575	4945.132
F71	Mean	1117.8857079625100000	1258.1065536572400000	659.5351969346130000	959.3735119754180000	911.4640642691360000	990.8546718748010000	836.1411004458200000
	Std	311.0011859260640000	359.7382897536570000	98.5410511961986000	240.5568407069990000	238.3180009803040000	235.1014092849970000	128.9346234954740000
	Best	560.0000000000000000	660.0000000000000000	560.0001912324020000	660.0000000000000000	560.0000121795840000	660.0000000000000000	560.0000000000000000
	Runtime	3012.883	241.541	2728.060	1573.484	10891.124	1769.459	2972.618
F72	Mean	1094.8305116977000000	−7.159E + 49	915.4958100611630000	1133.7536009808600000	1075.5292326436900000	1094.6823697304900000	984.5106541514410000
	Std	121.3539576317800000	4.387E + 50	242.1993331983530000	42.1171260000361000	166.9355145236330000	87.9884000140656000	199.1563947691970000
	Best	660.0000000000000000	−133.9585340104890000	660.0006867770510000	1088.9543269392600000	660.0000000000020000	660.0000000000000000	660.0000000000000000
	Runtime	6363.267	290.334	2326.112	1730.723	9601.880	3854.148	10458.467
F73	Mean	1304.3661550124000000	1159.9280867973000000	830.2290165794410000	1167.9040488743800000	1070.4327462836400000	1105.2511774948600000	976.2273885425320000
	Std	262.1065863453340000	742.1215416320490000	60.2286903507069000	236.7325108248320000	203.0676662707430000	190.6172874229610000	160.1543461970300000
	Best	919.4683107913200000	−460.7504508023100000	785.1725102979490000	785.1725102979490000	785.1725102979480000	919.4683107913240000	785.1725102979480000
	Runtime	2165.640	238.261	2045.582	1580.067	7459.005	1901.540	4209.110
F74	Mean	500.0000000000000000	653.3355378428050000	460.0000000000020000	510.0000000000000000	493.3333333333340000	490.0000000000000000	460.0000000000000000
	Std	103.7237710925280000	302.5312999719650000	0.0000000000016493	113.7147065368360000	137.2973951415090000	91.5385729888094000	0.0000000000000000
	Best	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000
	Runtime	1811.980	165.962	1698.121	1366.710	3016.959	1410.399	1795.637
F75	Mean	1107.9038127876700000	1401.6553278264300000	930.4565414149210000	1072.9924659809200000	1258.5157766524700000	1074.3695435628600000	1063.7363787709700000
	Std	127.9566489362040000	253.2428066220210000	87.9959072391079000	2.2606058314671500	241.4024507676890000	2.8314182838917800	55.8479313799755000
	Best	1069.5511765775700000	1072.4973401423200000	862.4476004191700000	1068.5560012648600000	871.8607884176050000	1069.8723890709000000	856.8214538442850000
	Runtime	4060.091	214.580	2113.339	2951.018	5262.210	3410.902	4280.901

Table 9
Determining the algorithm that statistically provides the best solution for each benchmark problem used in Test 2 by utilizing two-sided Wilcoxon Signed -Rank Test ($\alpha = 0.05$).

Problem	PSO vs. BSA				CMAES vs. BSA				ABC vs. BSA				JDE vs. BSA				CLPSO vs. BSA				SADE vs. BSA			
	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner
F51	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=	1.00E+00	0	0	=
F52	1.00E+00	0	0	=	1.00E+00	0	0	=	1.72E−06	0	465	+	1.00E+00	0	1	=	1.73E−06	0	465	+	1.00E+00	0	0	=
F53	1.73E−06	0	465	+	2.56E−06	435	0	−	1.73E−06	0	465	+	9.10E−01	227	238	=	1.73E−06	0	465	+	1.73E−06	0	465	+
F54	1.00E+00	0	1	=	4.01E−05	0	253	+	1.73E−06	0	465	+	3.91E−03	0	45	+	1.73E−06	0	465	+	1.00E+00	0	0	=
F55	6.10E−05	120	0	−	6.10E−05	120	0	−	1.73E−06	0	465	+	5.21E−02	36	117	=	1.73E−06	0	465	+	4.99E−01	100.5	70.5	=
F56	1.96E−05	25	440	+	1.25E−01	3	12	=	2.60E−05	28	437	+	4.88E−04	81	10	−	1.73E−06	0	465	+	5.00E−01	2	4	=
F57	1.73E−06	0	465	+	9.77E−04	0	66	+	1.00E+00	0	0	=	3.91E−03	0	45	+	1.00E+00	0	0	=	1.00E+00	1	0	=
F58	3.82E−01	190	275	=	8.73E−03	360	105	−	2.22E−04	53	412	+	1.73E−06	0	465	+	3.11E−05	435	30	−	1.40E−02	113	352	+
F59	1.73E−06	0	465	+	1.73E−06	0	465	+	1.00E+00	0	0	=	1.25E−01	0	10	=	6.10E−05	0	120	+	1.00E+00	0	1	=
F60	3.72E−05	433	32	−	1.84E−02	108.5	326.5	+	2.60E−06	4	461	+	6.36E−01	255.5	209.5	=	3.38E−03	375	90	−	1.78E−04	367.5	38.5	−
F61	8.29E−01	222	243	=	5.29E−04	64	401	+	1.73E−06	0	465	+	4.28E−02	134	331	+	2.88E−06	5	460	+	2.07E−02	345	120	−
F62	1.73E−06	0	465	+	5.30E−01	202	263	=	1.73E−06	0	465	+	1.92E−06	1	464	+	3.85E−03	373	92	−	7.04E−01	214	251	=
F63	1.73E−06	0	465	+	1.73E−06	0	465	+	6.64E−04	67	398	+	1.73E−06	0	465	+	1.20E−03	75	390	+	2.60E−06	4	461	+
F64	6.89E−05	426	39	−	1.73E−06	0	465	+	5.75E−06	12	453	+	2.37E−05	27	438	+	8.77E−01	225	240	=	1.25E−02	354	111	−
F65	2.50E−06	0	435	+	1.69E−06	0	465	+	4.88E−04	0	78	+	2.28E−06	0	435	+	6.25E−02	0	15	=	8.73E−05	0	210	+
F66	4.53E−04	403	62	−	1.25E−04	46	419	+	3.18E−06	6	459	+	1.78E−01	298	167	=	1.36E−01	305	160	=	1.13E−05	446	19	−
F67	5.71E−04	400	65	−	2.88E−06	5	460	+	1.73E−06	0	465	+	5.17E−01	201	264	=	1.04E−03	73	392	+	7.71E−04	396	69	−
F68	9.11E−02	109	242	=	7.25E−05	34	401	+	2.07E−02	345	120	−	7.62E−03	78	300	+	3.88E−06	8	457	+	2.04E−04	16	260	+
F69	2.13E−02	111	324	+	6.59E−04	60	375	+	3.16E−02	337	128	−	1.28E−03	55	323	+	1.97E−05	25	440	+	1.20E−03	37	263	+
F70	6.45E−02	132	303	=	1.00E−03	58.5	347.5	+	6.83E−03	364	101	−	3.30E−03	74	332	+	2.60E−05	28	437	+	3.57E−03	23	167	+
F71	2.93E−04	44	362	+	1.43E−05	12.5	393.5	+	5.31E−05	429	36	−	1.50E−02	65	235	+	9.37E−02	140	295	=	3.93E−03	49.5	250.5	+
F72	3.62E−03	83	352	+	1.75E−02	117	348	+	6.00E−01	207	258	=	5.75E−06	12	453	+	1.11E−03	74	391	+	7.86E−02	147	318	=
F73	3.10E−05	30	435	+	3.00E−02	127	338	+	1.54E−05	442	23	−	5.29E−04	64	401	+	2.41E−02	104	302	+	2.39E−03	85	380	+
F74	1.25E−01	0	10	=	4.88E−04	0	78	+	6.33E−05	0	136	+	6.25E−02	0	15	=	2.50E−01	0	6	=	2.50E−01	0	6	=
F75	2.18E−02	344	121	−	2.05E−04	52	413	+	3.32E−04	407	58	−	5.29E−04	401	64	−	8.19E−05	41	424	+	3.38E−03	375	90	−
+/=/−	11/8/6				18/4/3				15/4/6				14/9/2				15/7/3				9/10/6			

Table 10
Basic statistics of the 30-solutions obtained by PSO, CMAES, ABC, JDE, CLPSO, SADE and BSA in Test 3 (Mean: mean-solution, Std: standard-deviation of mean-solution, Best: the best-solution, and Runtime: mean-runtime in seconds.)

Problem	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	BSA
F76	Mean	−13.2203755617559000	−21.3171999570171000	−17.5170916858141000	−21.5430772832813000	−19.9209205768060000	−21.5379609140015000	−21.6111376389414000
	Std	0.5798709816552560	1.5463989317857500	1.0248612171337800	0.4455309716616480	1.6154416249189800	0.2422824984227170	0.1762084436112900
	Best	−14.4896840113398000	−21.8393554367690000	−19.4515843592484000	−21.8423582185690000	−21.3868251574175000	−21.8329482804784000	−21.8421583397469000
	Runtime	4129.778	5186.541	1542.859	2790.687	8448.335	5580.867	9171.871
F77	Mean	1.2784214305863400	0.7679943173132820	1.2084563403548400	0.7759733575203070	1.0109674682943700	0.6769744602128100	0.7475683099370200
	Std	0.2102758274100530	0.1467390614120050	0.1392680583116830	0.1663392331433730	0.1034903310851030	0.1590872901679670	0.1423974717303640
	Best	0.8957307058619430	0.5022449332178230	0.8706253318213730	0.5508361284548580	0.7849828809480960	0.5000000000000000	0.5000000000000000
	Runtime	601.297	419.529	120.789	399.582	493.750	590.092	520.584
F78	Mean	11.7951855564855000	22.6218132365045000	2.9898268828962900	7.0123888014188600	2.6037989610653200	1.4852730354889200	0.2805362483968100
	Std	4.7316108830775200	5.4861306841905700	4.5109093696318400	6.4577455248267800	4.8708666906638500	3.8776382320194000	1.5365603144480500
	Best	0.0000000000000000	8.4160874519043000	0.0059934075469223	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Runtime	146.615	17.797	40.992	9.098	197.590	15.556	23.169

Table 11
Determining the algorithm that statistically provides the best solution for each benchmark problem used in Test 3 by utilizing two-sided Wilcoxon Signed- Rank Test ($\alpha = 0.05$).

Problem	PSO vs. BSA				CMAES vs. BSA				ABC vs. BSA				JDE vs. BSA				CLPSO vs. BSA				SADE vs. BSA			
	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner	p-value	T+	T−	winner
F76	1.73E−06	0	465	+	0.490798	199	266	=	1.73E−06	0	465	+	0.797098	220	245	=	1.73E−06	0	465	+	0.093676	151	314	=
F77	1.73E−06	0	465	+	0.893644	239	226	=	1.73E−06	0	465	+	0.614315	208	257	=	5.75E−06	12	453	+	0.115608	309	156	=
F78	2.56E−06	0	435	+	1.73E−06	0	465	+	1.64E−05	23	442	+	0.000292	0	153	+	0.002282	9	127	+	0.125	1	14	=
+/=/−	3/0/0				1/2/0				3/0/0				1/2/0				3/0/0				0/3/0			

Table 12
Multi-problem based statistical pairwise comparison of comparison algorithms and BSA. ($\alpha = 0.05$).

Algorithm vs. BSA	p-Value	T+	T−	Winner
PSO vs. BSA	4.406E−07	145	1286	BSA
CMAES vs. BSA	4.081E−10	150	2196	BSA
ABC vs. BSA	2.007E−02	415	911	BSA
JDE vs. BSA	5.526E−08	121.5	1418.5	BSA
CLPSO vs. BSA	1.320E−07	186	1584	BSA
SADE vs. BSA	2.607E−04	194	841	BSA

and the median of the solutions obtained by algorithm B for same benchmark problem', i.e. median (A) = median (B). To determine whether algorithm A reached a statistically better solution than algorithm B, or if not, whether the alternative hypothesis was valid, the sizes of the ranks provided by the Wilcoxon Signed-Rank Test (i.e. T + and T− as defined in [46]) were examined.

3.6. Arithmetic precision

The level of arithmetic precision of numerous modern software development tools is 10^{-16} in the double-precision mode. Arithmetic precision value that is larger than necessary makes it difficult to compare the local search abilities of algorithms. Therefore, the arithmetic precision value used for the statistical tests in this paper was 10^{-16} to cover the precision level needed in many practical applications.

3.7. Statistical results of tests

Table 6 shows the mean runtimes and simple statistical values for the results obtained in Test 1. Table 7 lists the algorithms that obtained statistically better solutions compared with the other algorithms in Test 1, based on the Wilcoxon Signed-Rank Test.

Table 8 shows the mean runtimes and simple statistical values for the results obtained in Test 2. Table 9 lists the algorithms that provided statistically better solutions compared with the other algorithms in Test 2, based on the Wilcoxon Signed-Rank Test.

Table 10 shows the mean runtimes and simple statistical values for the results obtained in Test 3. Table 11 lists the algorithms that provided statistically better solutions compared with the other algorithms in Test 3, based on the Wilcoxon Signed-Rank Test.

Table 12 presents the multi-problem-based pairwise statistical comparison results using the averages of the global minimum values obtained through 30 runs of BSA and the comparison algorithms to solve the benchmark problems in Tests 1 and 2. These results show that BSA was statistically more successful than all of the comparison algorithms, with a statistical significance value $\alpha = 0.05$.

An examination of the results obtained from the tests made reveals that the success of BSA in solving the numerical optimization problems is generally not oversensitive to the problem dimension or the problem type.

In Tables 7,9 and 11, '+' indicates cases in which the null hypothesis was rejected and BSA displayed a statistically superior performance in the problem-based statistical comparison tests at the 95% significance level ($\alpha = 0.05$); '−' indicates cases in which the null hypothesis was rejected and BSA displayed an inferior performance; and '=' indicates cases in which there was no statistical difference between the two algorithms' success in solving the problems. The last rows of Tables 7,9 and 11 show the total counts in the (+/=/−) format for the three statistical significance cases (marked with '+', '=' or '−') in the pairwise comparison.

When the (+/=/−) values are examined, it can be said that BSA is statistically more successful than all the other comparison algorithms in solving the problems used in the Tests 1 and 2. When the (+/=/−) values for the Test 3 are examined, although the successes BSA and SADE have had are statistically identical, BSA has provided statistically better solutions than the other comparison algorithms.

4. Conclusions

This paper has introduced BSA, a new evolutionary-computing-based global search algorithm. BSA's algorithmic structure enables it to benefit from previous generation populations by using solutions it has found in the past for a given problem as it searches for solutions with better fitness values. BSA's bio-inspired philosophy is analogous to the return of a social group of living creatures at random intervals to hunting areas that were previously found fruitful for obtaining nourishment.

This paper presented three tests to examine BSA's success in solving numerical optimization problems by using 78 completely different benchmark problems. BSA's success in solving the numerical optimization problems was compared with that of several EAs that are widely used in the literature by using the Wilcoxon Signed-Rank Test.

BSA's success in solving real-world problems was examined in detail through three constrained benchmark problems (Antenna, Radar and FM) solved by BSA and comparison algorithms. The results indicate that BSA is generally more successful at solving problems than the comparison algorithms. For the Antenna, Radar and FM problems, BSA obtained statistically better solutions than ABC, JDE, CMAES, CLPSO and PSO2011. Similarly, BSA was statistically more successful than all of the comparison algorithms in solving the classical problems in Test 1. In solving the CEC2005 benchmark problems in Test 2, which are relatively more complex than the benchmark problems used in Test 1, BSA obtained much more successful results than the comparison algorithms. Moreover, an examination of the data obtained from the tests shows that BSA is generally faster than most of the comparison algorithms. The detailed tests discussed in this paper demonstrate that BSA is statistically successful in solving real-valued numerical optimization problems.

The factors responsible for BSA's greater success relative to the comparison algorithms are as follows:

- BSA's mutation and crossover operators produce very efficient trial populations in each generation.
- BSA's generation strategy for the parameter F, which controls the amplitude of the search direction, can produce both numerically large amplitude values necessary for a global search and the small amplitude values necessary for a local search in a very balanced and efficient manner. This clearly enhances BSA's problem-solving ability.
- The historical population (*oldP*) that BSA uses for the calculation of the search-direction matrix belongs to a randomly selected previous generation. Thus, the historical populations used in more advanced generations include more efficient individuals relative to the historical populations used in older generations. This facilitates BSA's generation of more efficient trial individuals.
- BSA's crossover strategy has a non-uniform and complex structure that ensures creation of new trial individuals in each generation. This crossover strategy enhances BSA's problem-solving ability.
- BSA's boundary control mechanism is very effective in achieving population diversity, which ensures efficient searches, even in advanced generations.

The problem-based pairwise statistical comparisons for the solutions obtained in Test 1 (see Table 7) show that BSA can solve a greater number of benchmark problems and can achieve statistically better results than the comparison algorithms. In Test 1, BSA was more successful in solving 22 problems compared with PSO, 28 problems compared with CMAES, 31 problems compared with ABC, 16 problems compared with JDE, 26 problems compared with CLPSO and 10 problems compared with SADE. In general, BSA's problem-solving success did not show any sensitivity to either the dimensions or the types (multimodal, non-separable/separable or unimodal) of the benchmark problems used in Test 1. In Test 1, the second most successful algorithm after BSA was SADE.

The problem-based pairwise statistical comparisons for the solutions obtained in Test 2 (see Table 9) show that the algorithms closest to BSA in terms of problem-solving success are in the order of JDE, CLPSO, CMAES, SADE, ABC and PSO. BSA obtained statistically better or statistically identical results in solving 23 problems compared with JDE, 22 problems compared with CLPSO and CMAES and 19 problems compared with SADE, ABC and PSO. On the other hand, SADE, ABC and PSO only obtained statistically better results than BSA in solving six benchmark problems, CLPSO and CMAES in solving three problems and JDE in solving two problems. The statistical analysis results in Table 9 show that BSA's problem-solving success is not significantly sensitive to the type of benchmark problem used in Test 2. Furthermore, BSA was able to solve more benchmark problems in Test 2 with statistically better results compared with the other algorithms.

For the multi-problem-based pairwise statistical comparison of BSA and the other algorithms, the average values of the solutions obtained in Tests 1 and 2 were used. Table 12 shows the $p-value$ and $T+$ and T− values obtained in this comparison. Analysis of these values when $\alpha = 0.05$ shows that BSA was statistically more successful than all of the comparison algorithms.

Acknowledgments

The author would like to thank the referees for suggestions that improved this paper's technical content. The studies in this paper were carried out under scientific research project #110Y309 supported by TUBITAK.

References

[1] K. Deb, A. Pratap, S. Agarwal, A fast and elitist multiobjective genetic algorithm: Nsga-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.
[2] H. Ishibuch, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput. 7 (2003) 204–223.
[3] J.K. Kishore, L.M. Patnaik, V. Mani, et al, Application of genetic programming for multicategory pattern classification, IEEE Trans. Evol. Comput. 4 (2000) 242–258.
[4] M.G. de Carvalho, A.H.F. Laender, M.A. Goncalves, et al, A genetic programming approach to record deduplication, IEEE Trans. Knowl. Data. Eng. 24 (2012) 399–412.
[5] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[6] 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 Trans. Evol. Comput. 10 (2006) 646–657.
[7] M.F. Tasgetiren, P.N. Suganthan, Q.K. Pan, An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem, Appl. Math. Comput. 215 (2010) 3356–3368.

[8] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[9] C. Kanzow, N. Yamashita, T. Fukushima, Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints, J. Comput. Appl. Math. 172 (2004) 375–397.
[10] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[11] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (2010) 61–106.
[12] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.
[13] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global. Optim. 39 (2007) 459–471.
[14] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99.
[15] A.A. Salman, I. Ahmad, M.G.H. Omran, et al, Frequency assignment problem in satellite communications using differential evolution, Comput. Oper. Res. 37 (2010) 2152–2163.
[16] I. De Falco, A. Della Cioppa, D. Maisto, et al, Differential evolution as a viable tool for satellite image registration, Appl. Softw. Comput. 8 (2008) 1453–1462.
[17] N. Najkar, F. Razzazi, H. Sameti, A novel approach to HMM-based speech recognition systems using particle swarm optimization, Math. Comput. Model. 52 (2010) 1910–1920.
[18] S. Wang, S. Wang, J.J. Ma, An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment, Sensors 7 (2007) 354–370.
[19] T. Sousa, A. Silva, A. Neves, Particle swarm based data mining algorithms for classification tasks, Parallel Comput. 30 (2004) 767–783.
[20] S. Das, A. Konar, A swarm intelligence approach to the synthesis of two-dimensional IIR filters, Eng. Appl. Artif. Intell. 20 (2007) 1086–1096.
[21] M.A. Mohandes, Modeling global solar radiation using Particle Swarm Optimization (PSO), Sol. Energy 86 (2012) 3137–3145.
[22] H. Qin, Aberration correction of a single aspheric lens with particle swarm algorithm, Opt. Commun. 285 (2012) 2996–3000.
[23] C.O. Ourique, E.C. Biscaia, J.C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Comput. Chem. Eng. 26 (2002) 1783–1793.
[24] A. Bonilla-Petriciolet, J.G. Segovia-Hernandez, Particle swarm optimization for phase stability and equilibrium calculations in reactive systems, Comput. Aid. Ch. 26 (2009) 635–640.
[25] B. Liu, L. Wang, Y. Liu, B. Qian, Y.H. Jin, An effective hybrid particle swarm optimization for batch scheduling of polypropylene processes, Comput. Aid. Ch. 34 (2010) 518–528.
[26] J. Zhang, L. Xie, S. Wang, Particle swarm for the dynamic optimization of biochemical processes, Comput. Aid. Ch. 21 (2006) 497–502.
[27] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man. Cybern. B 26 (1996) 29–41.
[28] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[29] X.S. Yang, S. Deb, Cuckoo search via levy flights World Congress on Nature and Biologically Inspired Computing'Nabic-2009, Coimbatore, India, vol. 4, 2009, pp. 210–214.
[30] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artif. Intell. Rev., in press. doi: http://dx.doi.org/10.1007/s10462-011-9276-0.
[31] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 13 (2009) 2232–2248.
[32] T. Xiang, X. Liao, K. Wong, An improved particle swarm optimization algorithm combined with piecewise linear chaotic map, Appl. Math. Comput. 190 (2007) 1637–1645.
[33] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73.
[34] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[35] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, IEEE Trans. Evol. Comput. 1–3 (2005) 1785–1791.
[36] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.
[37] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evol. Comput. 15 (2007) 1–28.
[38] R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: hybridization perspectives and experimental illustrations, Appl. Math. Comput. 217 (2011) 5208–5226.
[39] I.G. Tsoulos, A. Stavrakoudis, Enhancing PSO methods for global optimization, Appl. Math. Comput. 216 (2010) 2988–3001.
[40] R. Thangaraj, M. Pant, A. Abraham, et al, Particle swarm optimization: hybridization perspectives and experimental illustrations, Appl. Math. Comput 217 (2011) 5208–5226.
[41] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: IEEE Swarm Intelligence Symposium, Honolulu, 2007, 1-4244-0708-7.
[42] M.G.H. Omran, M. Clerc, 2011, <http://www.particleswarm.info/>, accessed 15 January, 2013.
[43] 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, 2005, pp. 1–50.
[44] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Technical Report, 2012, 1–42.
[45] P. Civicioglu, 2013, <http://www.pinarcivicioglu.com/bsa.html>, accessed 15 January, 2013.
[46] 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 Evol. Comput. 1 (2011) 3–18.