electronics

MDPI

Article

Heterogeneous Cooperative Bare-Bones Particle Swarm Optimization with Jump for High-Dimensional Problems

Joonwoo Lee [1,2] and Won Kim [3,*]

[1] Department of Electrical Engineering, Kyungpook National University (KNU), Daegu 41566, Korea; jwl@knu.ac.kr
[2] Department of Robot and Smart System Engineering, Kyungpook National University (KNU), Daegu 41566, Korea
[3] Department of Computer Science, Woosong University, Daejeon 304606, Korea
* Correspondence: kimwon@wsu.ac.kr

Received: 12 July 2020; Accepted: 12 September 2020; Published: 21 September 2020

Abstract: This paper proposes a novel Bare-Bones Particle Swarm Optimization (BBPSO) algorithm for solving high-dimensional problems. BBPSO is a variant of Particle Swarm Optimization (PSO) and is based on a Gaussian distribution. The BBPSO algorithm does not consider the selection of controllable parameters for PSO and is a simple but powerful optimization method. This algorithm, however, is vulnerable to high-dimensional problems, i.e., it easily becomes stuck at local optima and is subject to the "two steps forward, one step backward" phenomenon. This study improves its performance for high-dimensional problems by combining heterogeneous cooperation based on the exchange of information between particles to overcome the "two steps forward, one step backward" phenomenon and a jumping strategy to avoid local optima. The CEC 2010 Special Session on Large-Scale Global Optimization (LSGO) identified 20 benchmark problems that provide convenience and flexibility for comparing various optimization algorithms specifically designed for LSGO. Simulations are performed using these benchmark problems to verify the performance of the proposed optimizer by comparing the results of other variants of the PSO algorithm.

Keywords: bare-bones PSO (BBPSO); cooperative PSO (CPSO); high-dimensional optimization; large-scale global optimization; particle swarm optimization (PSO)

1. Introduction

Many optimization problems in modern engineering, e.g., optimal design and scheduling problems, must be solved with finite resources that should be used efficiently. In particular, most of these problems are high-dimensional and complex [1–3]. Therefore, the recent focus of optimization techniques has been on solving complex and high-dimensional problems, as described in [4–8].

The Particle Swarm Optimization (PSO) algorithm [9,10] is a metaheuristic inspired by the social behavior of birds flocking or fish schooling; the algorithm was created by simplifying this social behavior. In the algorithm, the so-called particles find a population of candidate solutions to a given optimization problem by moving in a search space according to simple mathematical formulas that are related to the particles' positions and velocities. The PSO algorithm can search the solution spaces of optimization problems with few or no assumptions about the problems, even those that involve searching relatively large spaces. Additionally, to be solved using the PSO algorithm, a problem need not be differentiable, and PSO can be robustly used with problems that include uncertainties such as noise or changes over time. Therefore, PSO algorithms are widely and frequently used to solve optimization problems because they are simple to implement, stable, and high-performing.

There have been many studies of PSO, which has great utility. Among these studies, there are several investigations into improving the convergence of the PSO algorithm and selecting the optimal PSO parameters [11–14]. Generally, the standard PSO algorithm has four primary user-controlled parameters. The selection of these parameters is known to have a considerable influence on the optimization algorithm's performance. PSO typically uses a uniform distribution to generate random numbers. However, the authors of [15,16] have demonstrated that escaping local optima is difficult for this PSO algorithm. Therefore, there have been many studies on PSO variants that use Gaussian distributions [16–18], which can reduce or remove some of the parameters. Of these algorithms, the Bare-Bones PSO (BBPSO) algorithm [19] is the simplest, and it can be intuitively understood and easily implemented without considering the parameter settings by sampling new particle positions from a Gaussian distribution whose mean is given by the average of the globally and locally best positions and whose standard deviation is given by the distance between the globally and locally best positions. Therefore, the BBPSO algorithm has attracted much interest from researchers. This algorithm, however, can become stuck at local optima when solving high-dimensional problems with many local optima, as mentioned in [17], and is subject to the "two steps forward, one step backward" phenomenon. This paper attempts to solve these problems and proposes an effective BBPSO-based optimizer for high-dimensional problems that combines heterogeneous cooperation based on the exchange of information between particles and a jumping strategy to avoid local optima. Figure 1 shows a brief summary of the goal and main idea for the development of the proposed Heterogeneous Cooperative BBPSO with Jumping (HCBBPSO-Jx) algorithms.

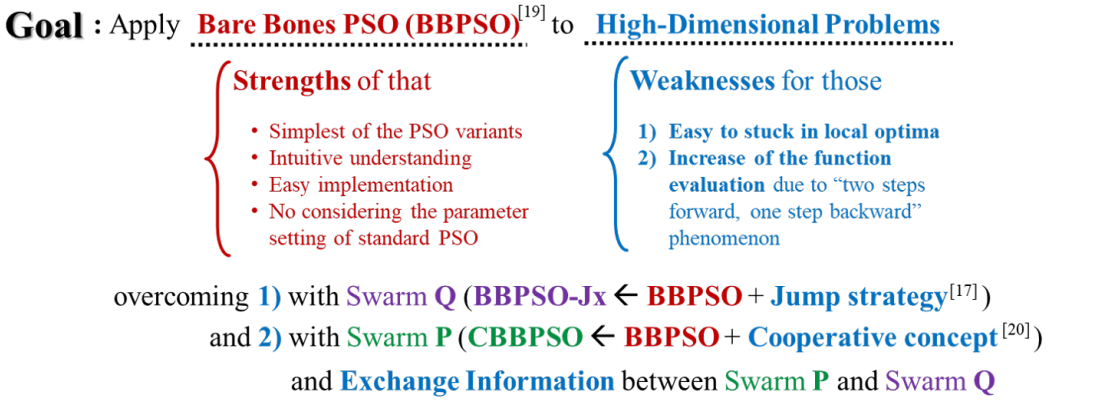


Figure 1. The goal of the proposed Heterogeneous Cooperative Bare-Bones Particle Swarm Optimization with Jumping (HCBBPSO-Jx) algorithms: to apply the simplest and powerful BBPSO [19] to high-dimensional problems, overcoming several weaknesses with jump strategy [17], cooperative concept [20], and exchange information between heterogeneous swarms.

The remainder of this paper is organized as follows: In Section 2, detailed mathematical models of the standard PSO and BBPSO algorithms are briefly introduced, and the cooperative learning and jumping strategies are described. Section 3 provides a detailed explanation of the proposed HCBBPSO-Jx algorithms. In Section 4, we verify the performance of the HCBBPSO-Jx algorithms using the 20 benchmark functions provided by the IEEE Congress on Evolutionary Computation 2010 (CEC 2010) Special Session on Large-Scale Global Optimization (LSGO) to provide convenience and flexibility for comparing various optimization algorithms. Using these large-scale global optimization problems, we compare the results with those of the other PSO variants. Finally, Section 5 concludes this paper.

2. Background Knowledge

2.1. The Particle Swarm Optimization (PSO) Algorithm

This section describes the mathematical model and the search procedure of the standard PSO algorithm. First, let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the cost or objective function that we should minimize. A candidate solution takes the form of a vector of real numbers, and the output of the function is a real number, which is the value of the objective function for the given candidate solution. The goal of this optimization problem is to find a solution $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in the search space, which is bounded by the values $\mathbf{b}_l$ and $\mathbf{b}_u$.

Algorithm 1 shows the process of the standard PSO algorithm [9,10]. The parameter N_P is the size of the population (called a swarm in the PSO algorithm) or the number of particles. As mentioned above, the parameters w, φ_p, and φ_g represent the inertia weight and the acceleration constants, respectively. Each particle has a position $\vec{x}_i(k) \in \mathbb{R}^n$ in the search space and a velocity $\vec{v}_i(k) \in \mathbb{R}^n$, at time k. The vector $\vec{p}_i(k)$ is the best known position of particle i, and the vector $\vec{g}(k)$ is the best known position of the entire swarm at time k. These are also called "p_{best}" and "g_{best}," respectively.

Before the algorithm begins, each particle's position and velocity are initialized with uniformly distributed random vectors, i.e., $\vec{x}_i(1) \sim U(\mathbf{b}_l, \mathbf{b}_u)$ and $\vec{v}_i(1) \sim U(-|\mathbf{b}_u - \mathbf{b}_l|, |\mathbf{b}_u - \mathbf{b}_l|)$, respectively. The vector $\vec{p}_i(1)$ is first initialized with the vector $\vec{x}_i(1)$. Finally, the vector $\vec{g}(1)$ is initialized with the best of the vectors $\mathbf{p}(1)$.

Algorithm 1 Standard PSO Algorithm.

```
SetParameters(N_P, w, φ_p, φ_g);                    ▷ N_P: # of Particles
Initialize(x(1), p(1), g(1), v(1));
k = 1;
while Termination condition does not meet do
    for i = 1 to N_P do
        for d = 1 to n do                           ▷ n: Dimension of Problem
            r_p, r_g ~ U(0,1);
            k = k + 1;
            Update(v_{i,d}(k));                     ▷ Using Equation (1)
        end for
        Update(x_i(k));                             ▷ Using Equation (2)
        if f(x_i(k)) < f(p_i(k)) then
            p_i(k) ← x_i(k);                        ▷ Update p_best
            if f(p_i(k)) < f(g(k)) then
                g(k) ← p_i(k);                      ▷ Update g_best
            end if
        end if
    end for
end while
return g(k);                                        ▷ Best Found Solution
```

Next, the particles search begins. For each dimension of each particle, an element $v_{i,d}$ is determined as follows:

$$v_{i,d}(k+1) = wv_{i,d}(k) + \varphi_p r_p \left(p_{i,d}(k) - x_{i,d}(k)\right) + \varphi_g r_g \left(g_d(k) - x_{i,d}(k)\right), \quad (1)$$

where the parameters r_p and r_g are uniformly distributed random numbers in the range $[0,\ 1]$, i.e., $U(0,\ 1)$. In the first part of Equation (1), the inertial weight w represents the amount of momentum

the particles have. The second part is the "cognition" part, which represents the independent behavior of each particle. The final part is the "social" part, which represents the collaboration among the particles. The constants φ_p and φ_g determine the relative influences of the cognition and social parts, and eventually, pull each particle toward position p_{best} and g_{best}.

The next position of particle i, $\vec{x}_i(k+1)$, is calculated using Equation (1) as follows:

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1). \tag{2}$$

Subsequently, if the value of $f(\vec{x}_i(k))$ is smaller than the value of $f(\vec{p}_i(k))$, then the vector $\vec{p}_i(k)$ is updated to $\vec{x}_i(k)$. In addition, if the value of $f(\vec{p}_i(k))$ is smaller than the value of $f(\vec{g}(k))$, then, the vector $\vec{g}(k)$ is updated to $\vec{p}_i(k)$.

This process is repeated until a termination criterion is met, e.g., the maximum number of iterations or a solution with an acceptable objective function value is reached.

2.2. The Bare-Bones PSO (BBPSO) Algorithm

As mentioned above, the BBPSO algorithm does not have to set up the parameters w, φ_p, and φ_g, unlike the standard PSO algorithm. In the BBPSO algorithm, the next position of a particle is determined by sampling a Gaussian distribution whose mean is given by the average globally best position of the swarm(s), g_{best}, and the personally-best position of the particle, p_{best} and whose the standard deviation given by the absolute difference between g_{best} and p_{best}. For each element (or dimension) of a particle, the next position is determined in the BBPSO algorithm using the following equations instead of using Equations (1) and (2), which are for the standard PSO algorithm:

$$x_{i,d}(k+1) = N\left(\mu_{i,d}(k), \sigma^2_{i,d}(k)\right) \tag{3}$$

$$\mu_{i,d}(k) = \frac{g_d(k) + p_{i,d}(k)}{2} \tag{4}$$

$$\sigma_{i,d}(k) = \left|g_d(k) - p_{i,d}(k)\right|, \tag{5}$$

where $N\left(\mu_{i,d}(k), \sigma^2_{i,d}(k)\right)$ is a random number generator based on a Gaussian distribution with mean $\mu_{i,d}(k)$ in Equation (4) and standard deviation $\sigma_{i,d}(k)$ in Equation (5) for the d-th dimension of the particle i. Except for the above step, the procedure is equivalent to that of the standard PSO algorithm.

2.3. The Cooperative Approach

Generally, in population-based algorithms, including the PSO algorithm, an agent in one population represents an intact n-dimensional candidate solution. In the standard PSO algorithm, there is one swarm of N_P particles, each of which has n components, that attempts to find an optimal n-dimensional solution. However, in this case, the algorithm frequently undergoes the "two steps forward, one step backward" phenomenon described in [20], especially when it is solving a high-dimensional problem. This appearance of this phenomenon means that although the fitness of a particle (or a candidate solution vector) may be considerably improved during the next time step, some of its components may have been changed from a better value to a rather poor value. Indeed, an improvement in two components (two steps forward) overrules a potentially good value for a single component (one step backward) for a problem with a three-dimensional solution vector. Eventually, valuable information is unintentionally lost.

One solution to the "two steps forward, one step backward" problem is to evaluate the objective function more frequently, perhaps each time a component in the candidate solution vector is updated, which results in much quicker feedback. However, in this case a problem remains. The function is only evaluated for a complete n-dimensional solution vector. Therefore, after a specific component is updated, the values of the $n-1$ other components of the candidate vector still need to be

chosen. One method of overcoming these problems proposed in [20], the Cooperative PSO (CPSO) algorithm, employs a cooperative approach. In the CPSO algorithm, unlike the standard PSO algorithm, the solution vector is split into its components so that K swarms of N_P particles containing $\lceil n/K \rceil$-dimensional or $\lfloor n/K \rfloor$-dimensional components, where K is a pre-determined parameter called the split factor, are optimized. This approach effectively increases the solution diversity and the amount of information exchanged to avoid the "two steps forward, one step backward" phenomenon.

2.4. The Jumping Strategy

When variants of the PSO algorithm are applied to optimization problems with many local optima in a high-dimensional search space, they may become stuck at the local optima. The jumping strategy [17,21–23] was proposed to escape from local optima, and promising results have been obtained. This strategy has been implemented as a mutation operator in Evolutionary Algorithms (EAs) based on a Gaussian and Cauchy probability distributions.

The goal of the jumping strategy is to allow particles in PSO algorithms to escape from local optima to which they are prematurely attracted. The motion of the particles in this situation stagnates with no improvement in their fitness. Whether a particle is stagnating can be determined by monitoring its fitness, and then, the stagnating particles move to a new point that is selected using the jumping strategy. This aim can be accomplished by introducing a *stagnation interval* ($C_{F,j}$ for each particle in this paper), which monitors the fitness of each particle, and increasing it by one during each iteration until its value reaches a pre-determined maximum number of iterations, which is called the maximum stagnation interval in [17] (M_F in this paper).

When the particle should jump to a new point, its next position is determined by choosing between Gaussian and Cauchy jumps as follows:

$$\vec{x}_i(k+1) = \vec{p}_i(k) \cdot (1 + \eta \cdot N(0,1)) \tag{6}$$

$$\vec{x}_i(k+1) = \vec{p}_i(k) \cdot (1 + \eta \cdot C(0,1)) \tag{7}$$

where the parameter η is for scaling and the vector $\vec{p}_i(k)$ is the best known position of particle i. In Equation (6), $N(0,1)$ is a random number generated using a Gaussian probability distribution with a mean of 0 and a standard deviation of 1. In Equation (7), $C(0,\ 1)$ is a random number generated using a Cauchy probability distribution with $\gamma = 1$ centered at the origin and described by

$$f(x) = \frac{\gamma}{\pi\,(\gamma^2 + x^2)}, \quad -\infty < x < \infty. \tag{8}$$

3. Heterogeneous Cooperative BBPSO with Jumping (HCBBPSO-Jx) Algorithms

Algorithm 2 shows detailed pseudo-code for the Heterogeneous Cooperative BBPSO with the jumping (HCBBPSO-Jx) algorithms proposed in this paper. The proposed algorithms consist of three main parts: a cooperative BBPSO step, a BBPSO with jumping step, and a cooperative part in which information is exchanged between two heterogeneous algorithms. First, the parameters for the HCBBPSO-Jx algorithms are initialized. The matrices **x** and **y** contain the current position vectors of the particles in swarms **P** and swarm **Q**, respectively. Additionally, the matrices **p** and **q** (the bottom matrices in Figure 2) include the p_{best} information of swarms **P** and swarm **Q**, respectively. The vector $\vec{g}_P$ (the top vectors in Figure 2) stores the g_{best} information of swarms **P** and the vector $\vec{g}_Q$ stores that of swarm **Q**.

3.1. The Cooperative BBPSO (CBBPSO) Step

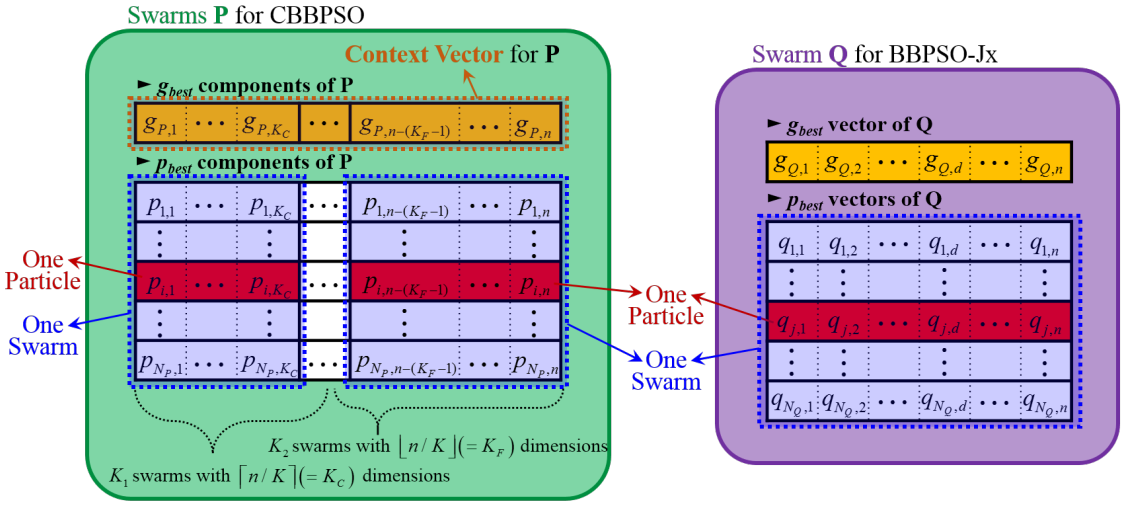
Once the necessary parameters for the HCBBPSO-Jx algorithms have been initialized, the Cooperative BBPSO (CBBPSO) step is performed. This step introduces the aforementioned cooperative approach. It has been modified for use in high-dimensional problems as shown in lines 7–18 of Algorithm 2. Unlike the CPSO algorithm [20], the HCBBPSO-Jx algorithms are based on

the BBPSO algorithm instead of the traditional PSO algorithm, i.e., they use the equation of motion from the BBPSO algorithm, Equation (3), to move the particles. This method is simpler and more robust than that of the PSO algorithm. Additionally, the algorithm uses the **P** swarms, which are K swarms of N_P particles, as shown in Figure 2a. Here, the constant K is a pre-determined parameter called the split factor, as it is in the CPSO algorithm. From the constant K, the parameter K_1 is calculated as $K_1 = mod(n,\ K)$. Then, the parameter K_2 is $K - K_1$. Of the K **P** swarms, K_1 contain particles that have K_C dimensions, where $K_C = \lceil n/K \rceil$. The particles in the remaining K_2 swarms have K_F-dimensional components, where $K_F = \lfloor n/K \rfloor$. That is to say, an n-dimensional solution vector is divided into K_1 K_C-dimensional components and K_2 K_F-dimensional components, where $n = K_C \times K_1 + K_F \times K_2$. In addition, to reduce the Number of Function Evaluations (NFEs), the proposed algorithm uses a double if statement for updates (lines 10–15 of Algorithm 2).

As in [20], for cooperation (or, more precisely, information exchange) between swarms, a "blackboard," which is a shared memory in which particles can post or read hints, is used. To establish this blackboard, the algorithm introduces a context vector, shown at the top of Figure 2a, which selects the globally best particle from each of the K **P** swarms, and is used to evaluate the particles. To evaluate all of the particles in the s-th swarm, the other $n - K_X$ components in the context vector are kept constant while the $1 + (s-1)K_X$-th to the sK_X-th components of the context vector are replaced by each particle from the s-th swarm in turn. The function $\vec{B}(s, z_s)$ shown (and in lines 10 and 12 of Algorithm 2) plays this role to create an n-dimensional vector that is evaluated.

$$\vec{B}(s, z_s) \equiv \left(g_{P,1},\ \cdots,\ g_{P,(s-1)K_X},\ z_s,\ g_{P,sK_X},\ \cdots,\ g_{P,n}\right) \tag{9}$$

The subscript X of K_X is determined as follows: for the s-th swarm, if $s \in \{1,\ \cdots, K_1\}$ then $X = C$ and $X = F$ if $s \in \{K_1 + 1,\ \cdots, K\}$.



(a) Swarms **P** for Cooperative BBPSO (CBBPSO) (b) Swarm **Q** for BBPSO with Jump (BBPSO-Jx)

Figure 2. The configurations of g_{best} and p_{best} for the HCBBPSO-Jx algorithms: (**a**) swarms **P**: K swarms of N_P particles; in the cooperative BBPSO (CBBPSO) step, the **P** swarms consist of K_1 $\lceil n/K \rceil$-dimensional swarms and K_2 $\lfloor n/k \rfloor$-dimensional swarms, i.e., $K = K_1 + K_2$ where $K_1 = mod(n,\ K)$; (**b**) swarm **Q**: an n-dimensional swarm of N_Q particles for the BBPSO algorithms with Jumping (BBPSO-Jx) (one particle = one row vector).

3.2. The BBPSO with Jumping (BBPSO-Jx) Step

As the second step, to achieve robust heterogeneous cooperation for solving high-dimensional problems, the proposed HCBBPSO-Jx algorithms introduce the BBPSO algorithm with a jumping strategy. Figure 2b displays the configuration of the g_{best} and p_{best} of the swarm **Q**, which is the general form

of the PSO variants. The standard BBPSO algorithm may become stuck at a local optimum when solving a high-dimensional problem with many local optima. In this case, as mentioned in Section 2.4, the jumping strategy enables particles to escape from the local optima. If there is no improvement for the j-th particle, then, the update failure counter, $C_{F,j}$, is increased by one during each iteration until it reaches the predefined maximum allowable number of update failures, M_F. After reaching the maximum number of failures, M_F, the particle jumps to a new point using Equation (6) or (7). After the performances of two cases, which use Gaussian and Cauchy probability distributions, have been compared, the best jumping strategy for the proposed HCBBPSO-Jx algorithms is identified in Section 4.

Algorithm 2 HCBBPSO-Jx Algorithms (for x, C: Cauchy, G: Gaussian).

SetParameters(N_P, K, N_Q, η, M_F); ▷ **Set parameters**
▷ **K: split factor, M_F: # of maximum allowable update failure, η: jump scaling factor, n: dimension of given problem**
$k = 1$;
$K_1 = mod(n, K)$; $K_2 = K - K_1$; $K_C = ceil(n/K)$; $K_F = floor(n/K)$;
▷ **For spliting into solution vector components for CBBPSO**
Initialize($\mathbf{x}(1)$, $\mathbf{p}(1)$, $\vec{g}_P(1)$, $\mathbf{y}(1)$, $\mathbf{q}(1)$, $\vec{g}_Q(1)$, $\vec{C}_F$);
▷ **Initialize swarms P and swarm Q , $\vec{C}_F$: Counter vector for update failure**
while Termination condition does not meet **do**
$k = k + 1$;
▼▼ **Cooperative BBPSO Step** ▼▼
for $s = 1$ **to** K **do** ▷ **For s-th swarm with K_C dimensions if $s \in \{1, \cdots, K_1\}$ and K_F dimensions if $s \in \{K_1 + 1, \cdots, K\}$**
$D = [1 + (s-1)K_X : sK_X]$; ▷ **$X = C$ if $s \in \{1, \cdots, K_1\}$ and $X = F$ if $s \in \{K_1 + 1, \cdots, K\}$**
for $i = 1$ **to** N_P **do** ▷ **For each particle**
if $f(\vec{B}(D, x_{i,D}(k))) < f(\vec{B}(D, p_{i,D}(k)))$ **then**
$p_{i,D}(k) \leftarrow x_{i,D}(k)$; ▷ **Update p_{best} components (or particle) of swarms P**
if $f(\vec{B}(D, p_{i,D}(k))) < f(\vec{B}(D, g_{P,D}(k)))$ **then**
$g_{P,D}(k) \leftarrow p_{i,D}(k)$; ▷ **Update g_{best} components of swarms P (K_C or K_F components of context vector)**
end if
end if
end for
$\vec{x}_D(k) \sim N(\vec{\mu}_D(k), \vec{\sigma}_D^2(k))$; ▷ **Moving swarm using Equations (3)–(5) for swarms P**
end for
▼▼ **BBPSO with Jump Step** ▼▼
for $j = 1$ **to** N_Q **do** ▷ **For each particle**
if $C_{F,j} \leq M_F$ **then** ▷ **Perform original BBPSO**
$\vec{y}_j(k) \sim N(\vec{\mu}_j(k), \vec{\sigma}_j^2(k))$; ▷ **Moving swarm using Equations (3)–(5) for swarm Q**
else ▷ **Perform jump**
$C_{F,j} \leftarrow 0$;
$\vec{y}_j(k) \sim \vec{q}_j(k) \cdot (1 + \eta \cdot C(0,1))$; ▷ **$C(0,1)$ for Cauchy jump or $N(0,1)$ for Gaussian jump**
end if
for $s = 1$ **to** K **do** ▷ **Information exchange from swarms P to swarm Q**
$D = [1 + (s-1)K_X : sK_X]$; ▷ **$X = C$ if $s \in \{1, \cdots, K_1\}$ and $X = F$ if $s \in \{K_1 + 1, \cdots, K\}$**
$m \sim U(1, N_Q)$; ▷ **Integer uniform distribution with range $[1, N_P]$**
$y_{m,D}(k) \leftarrow g_{P,D}(k)$;
end for
if $f(\vec{y}_j(k)) < f(\vec{q}_j(k))$ **then**
$\vec{q}_j(k) \leftarrow \vec{y}_j(k)$; ▷ **Update p_{best} vector (or particle) of swarm Q**
if $f(\vec{q}_j(k)) < f(\vec{g}_Q(k))$ **then**
$\vec{g}_Q(k) \leftarrow \vec{q}_j(k)$; ▷ **Update g_{best} vector of swarm Q**
end if
else
$C_{F,j} \leftarrow C_{F,j} + 1$;
end if
end for
for $d = 1$ **to** n **do** ▷ **Information exchange from swarm Q to swarms P**
$m \sim U(1, N_P)$; ▷ **Integer uniform distribution with range $[1, N_Q]$**
$x_{m,d}(k) \leftarrow g_{Q,d}(k)$;
end for
$\vec{g}(k) \leftarrow min(\vec{g}_P(k), \vec{g}_Q(k))$
end while
return $\vec{g}(k)$; ▷ **Best found solution**

3.3. The Steps Involving Cooperation by Exchanging Information between the CBBPSO and BBPSO-Jx Algorithms

The final part of the HCBBPSO-Jx algorithms is where heterogeneous cooperation between the CBBPSO step and the BBPSO with jumping step occurs in the HCBBPSO-Jx algorithms; information from the previous two steps is exchanged, as shown in lines 26–30 and 40–43 of Algorithm 2. In this step, information is exchanged once per iteration when n components that are randomly selected from the matrices $\mathbf{x}$ and $\mathbf{y}$, each corresponding to a component of its g_{best}, are substituted. This step helps increase the diversity of the solutions searched.

Like other variants of the PSO algorithm, the above three-step process is repeated until a termination criterion is met, e.g., the maximum number of iterations or a solution with an acceptable objective function value is reached.

4. Comparative Simulations

This section presents the results of the simulations and a discussion of comparable simulations performed using five variants of the PSO algorithm. The goal is to verify the performance of the proposed HCBBPSO-Jx algorithms by applying them to the 20 1000-dimensional benchmark functions from the CEC 2010 Special Session. The CEC 2010 Special Session on Large-Scale Global Optimization (LSGO) identified 20 benchmark problems [24] that provide convenience and flexibility for comparing various optimization algorithms that are specifically designed for large-scale global optimization. The test suite includes four types of high-dimensional problem: (1) separable functions; (2) partially separable functions, which have a small number of dependent variables, and all of the remaining variables are independent; (3) partially separable functions that consist of multiple independent subcomponents, each of which is m-nonseparable; and (4) fully nonseparable functions. The detailed mathematical formulas for and properties of these functions are described in [24]. Section 4.1 describes the simulation environment and setup. Section 4.2 describes the comparative results evaluated using the Formula One point system, which is the method used in the LSGO challenge posed in the CEC 2010 competition. Finally, Section 4.3 is dedicated to reporting the results of the best algorithm tested in the comparative simulations using the method that represented the results of the LSGO competition.

4.1. The Simulation Environment and Setup

The simulations were conducted using the 20 1000-dimensional minimization problems from CEC 2010. For each problem, the simulation was run 25 times for statistical accuracy. The simulation terminated when the maximum number of function evaluations (MaxNFEs), which was set to 3×10^6 for all of the trials, was reached. The simulator and algorithm used in this simulation were implemented in MATLAB for 32-bit Windows 8.1. All of the simulations were performed on four computers with an Intel @ 3.07 GHz processor and 4 GB of RAM. For fairness, each computer only ran simulations for the following pre-assigned functions: $f_1 \sim f_5$ on COM1, $f_6 \sim f_{10}$ on COM2, $f_{11} \sim f_{15}$ on COM3, and $f_{16} \sim f_{20}$ on COM4.

For comparison, the five variants of the PSO algorithm included the two groups of algorithms shown in Table 1. The first group included three algorithms that are associated with the proposed HCBBPSO-Jx algorithms: The cooperative PSO-H_K (CPSO-H_K) algorithm, which was the most robust of the CPSO variants proposed in [20] and was obtained by combining the CPSO-S_K algorithm with the PSO algorithm and Bare-Bones PSO with Cauchy and Gaussian jumping (BBPSOjumpC and BBPSOjumpG, respectively) algorithms [17], which improved the ability of the BBPSO algorithm to escape from local optima. The second comprised well-known variants of the PSO algorithm. The Adaptive PSO (APSO) algorithm [25] used a parameter adaptation scheme and elitist learning strategy to improve the PSO algorithm. The Comprehensive Learning PSO (CLPSO) algorithm [26] improved the diversification ability of the PSO algorithm by using comprehensive learning, in which all of the historically best information on the other particles was used to update a particle's velocity. All of the algorithms used the same parameters in all of the simulations; these are shown in the table.

The parameters for each algorithm were assigned the values that resulted in the best performance and that were recommended in the literature. In addition, to achieve a fair test, the initial population size was set to 50 for each algorithm. All of the search parameters were initialized using a uniform random process within the search space.

Table 1. Parameter setting of each algorithm for comparative simulation with Particle Swarm Optimization (PSO) variants.

Algorithm	Parameter	Value
HCBBPSO-Jx (x = C or G)	Number of particles in Swarms **P** (N_P)	25
	Split factor (K)	50
	Number of particles in Swarm **Q** (N_Q)	25
	Jump scaling factor (η)	1.1
	Number of maximum allowable update failure (M_F)	5
CPSO-H$_K$ [20]	Number of particles in each algorithm (CPSO-S$_K$ and PSO respectively)	25
	Split factor (K)	6
	Weight for previous velocity (w)	0.72
	Coefficient of cognition term (c_1)	1.49
	Coefficient of social term (c_2)	1.49
BBPSOjumpx [17] (x = C or G)	Number of particles	50
	Jump scaling factor (η)	1.1
	Number of maximum allowable update failure (M_F)	5
APSO [25]	Number of particles	50
	Weight for previous velocity	0.9
	Coefficient of cognition term	2
	Coefficient of social term	2
CLPSO [26]	Number of particles	50
	w_0	0.9
	w_1	0.4
	Coefficient of cognition term	1.49445
	Coefficient of social term	1.49445
	Refreshing gap (m)	7
	$V_{max} = -V_{min}$ (x_L^i, x_U^i: Lower and upper bound of i-th dimension)	$0.2(x_U^i - x_L^i)$

4.2. The Results of Comparing the Simulations Performed by the PSO Algorithm Variants

Table 2 shows the results of the simulations performed by the variants of the PSO algorithm after evaluation using the scoring system from the CEC 2010 LSGO Challenge. The scoring system used in the CEC 2010 LSGO Challenge was as follows: for each algorithm, a table of the type shown in Table 7 that contains 300 competition categories was formed. The competition categories were 20 functions ($f_1 \sim f_{20}$), 3 limits on the NFEs (1.2×10^5, 6.0×10^5, and 3.0×10^6), and 5 statistical values (best, median, worst, mean, and standard deviation) at each limit on the NFEs for each of the 25 runs. Then, the LSGO Challenge applies the Formula One point system to the data from the challenge participants in each of the 300 categories. Table 3 shows the points for each ranking in the Formula One point system. The winner received 25 points and other rankers received differentiated points according to ranking. Like the CEC 2010 LSGO Challenge, in this simulation, the smaller measured values in all of the categories, the higher the ranking and the more points. In particular, a small standard deviation means that the performance was more reliable. Eventually, the participant with the highest total score wins. In the results of the evaluation using this scoring system, the proposed HCBBPSO-JG algorithm was the best of the participating algorithms. In addition, the HCBBPSO-JG algorithm performed better than the HCBBPSO-JC algorithm that was proposed alongside it.

Table 2. The results of applying the scoring system from the CEC 2010 LSGO Challenge to the results of the simulations performed by variants of the PSO algorithm.

Ranking	**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**
Algorithm	**HCBBPSO-JG**	HCBBPSO-JC	BBPSOjumpC	BBPSOjumpG	CLPSO	$CPSO_6$	APSO
Total Score	**5474**	5357	4276	4233	4131	2517	2212

Table 3. The results of applying the scoring system from the CEC 2010 LSGO Challenge to the results of the simulations performed by variants of the PSO algorithm.

Ranking	**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**	**10th**
Points	25	18	15	12	10	8	6	4	2	1

Tables 4–6 provide the results of the 25 runs of each variant of the PSO algorithm used in the comparison process when the NFEs counter reached $= 3.0 \times 10^6$ for the 20 functions. We conducted several statistical hypothesis tests on these results. First, we performed the Friedman rank test on the data from all of the algorithms. Next, if there was a significant difference at the 5% significance level, we performed the Wilcoxon signed-rank test on the best algorithm, i.e., the HCBBPSO-JG algorithm, as well as the other algorithms and marked the results of each statistical significance test with its p-value and sign in the tables; the sign "$+$" means that the HCBBPSO-JG algorithm was significantly better than the algorithm compared to it, the sign "$\sim$" means that the two were not significantly different, and the sign "$-$" means that the HCBBPSO-JG algorithm was significantly worse than the algorithm compared to it. The results of the Friedman rank test show that there were significant differences in the data from all of the algorithms at the p-value $\approx 0 \ll 0.05$ significance level; therefore, the Wilcoxon signed-rank test was performed on the HCBBPSO-JG algorithm and the other algorithms. In the tables, the measured value of each function written in bold in colored cells represents the value of the best algorithm for each statistical value.

In these comparative results, the proposed HCBBPSO-Jx algorithms performed better than the other algorithms; they won for a total of 12 functions in terms of the mean NFEs, which is similar to the results obtained using the CEC 2010 LSGO Challenge scoring system. In particular, the HCBBPSO-JG algorithm stayed ahead of the HCBBPSO-JC algorithm with significant differences in the three functions f_7, f_{16}, and f_{17}. However, the HCBBPSO-Jx algorithms were the weakest of the variants of Rastrigin's function, i.e., f_2, f_5, f_{10}, and f_{15}. The cause of this phenomenon can be understood by examining the convergence curves shown in Figure 3. We know from the curves for f_2, f_5, f_{10}, and f_{15} that the HCBBPSO-JG algorithm prematurely converged to a local optimum only for the functions that were connected to Rastrigin's function. The first cause for this result was to simulate all algorithms using the same total number of particles in order to ensure fairness in comparison simulation. The total number of particles directly affects the increase or decrease of NFEs (Number of Function Evaluations) of an algorithm because one particle must use at least one function evaluation to be evaluated for the solution it finds itself. Therefore, it cannot be a fair comparison if simulations are performed with different particle size. The second reason for these results is that the optimal parameter value for the parameter split factor K, which governs the performance of the HCBBPSO-Jx algorithms, was not used. Selecting this parameter K is very hard because it depends on the problems that want to be solved. In particular, the problems to solve with the algorithms proposed in this paper are high-dimensional problems with high complexity. Therefore, the issue of selecting optimal parameter values for the proposed HCBBPSO-Jx algorithms is to be left behind for further in-depth research later as the further works of this paper.

Table 4. A comparison of the results of simulations performed by variants of the PSO algorithm. The best, median, worst, mean, standard deviation, *p*-value, and significance sign of the 25 runs when the NFEs counter reached 3.0×10^6 for the functions $f_1 \sim f_7$ are reported. The *p*-value was determined using the Wilcoxon signed-rank test between the best algorithm (HCBBPSO-JG) and the others. The significance sign "+" means that the HCBBPSO-JG algorithm was significantly better than the algorithm compared to it, the sign "∼" means that the two were not significantly different, and the sign "−" means that the HCBBPSO-JG algorithm was significantly worse than the algorithm compared to it. The measured value written in bold in colored cells for each function represents the value of the best algorithm for each statistical value.

Algorithm		f_1	f_2	f_3	f_4	f_5	f_6	f_7
APSO	**Best**	5.92×10^5	2.37×10^4	1.99×10^1	8.91×10^{12}	2.94×10^8	2.26×10^6	1.86×10^8
	Median	7.66×10^6	2.42×10^4	2.15×10^1	1.55×10^{13}	3.85×10^8	1.98×10^7	5.23×10^8
	Worst	3.63×10^{11}	2.44×10^4	2.15×10^1	2.96×10^{13}	6.55×10^8	2.11×10^7	1.38×10^9
	Mean	7.02×10^{10}	2.42×10^4	2.13×10^1	1.67×10^{13}	4.02×10^8	1.49×10^7	6.45×10^8
	Std	1.45×10^{11}	2.17×10^2	4.30×10^{-1}	5.40×10^{12}	9.82×10^7	8.31×10^6	3.68×10^8
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.54×10^{-4}	6.39×10^{-1}	6.10×10^{-5}
	(Sign)	(+)	(+)	(+)	(+)	(−)	(∼)	(+)
BBPSOjumpC	**Best**	1.24×10^6	4.21×10^3	1.93×10^1	1.49×10^{12}	9.45×10^7	1.03×10^6	7.77×10^4
	Median	9.58×10^6	4.50×10^3	1.95×10^1	2.71×10^{12}	1.44×10^8	2.20×10^6	1.67×10^5
	Worst	4.92×10^7	4.91×10^3	1.97×10^1	5.61×10^{12}	$\mathbf{2.02 \times 10^8}$	3.44×10^6	7.33×10^5
	Mean	1.42×10^7	4.54×10^3	1.95×10^1	3.21×10^{12}	1.45×10^8	2.19×10^6	2.24×10^5
	Std	1.51×10^7	2.26×10^2	1.08×10^{-1}	1.23×10^{12}	3.59×10^7	7.20×10^5	1.71×10^5
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	3.03×10^{-1}
	(Sign)	(+)	(+)	(+)	(+)	(−)	(−)	(∼)
BBPSOjumpG	**Best**	6.16×10^5	4.34×10^3	1.91×10^1	1.55×10^{12}	$\mathbf{8.96 \times 10^7}$	$\mathbf{1.99 \times 10^1}$	5.86×10^4
	Median	4.85×10^6	4.62×10^3	1.93×10^1	3.17×10^{12}	$\mathbf{1.30 \times 10^8}$	2.26×10^6	1.87×10^5
	Worst	5.05×10^7	4.89×10^3	1.95×10^1	7.85×10^{12}	2.13×10^8	4.11×10^6	5.92×10^5
	Mean	1.11×10^7	4.63×10^3	1.94×10^1	3.36×10^{12}	$\mathbf{1.41 \times 10^8}$	2.38×10^6	2.28×10^5
	Std	1.31×10^7	1.48×10^2	1.28×10^{-1}	1.56×10^{12}	4.24×10^7	9.76×10^5	1.60×10^5
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	7.30×10^{-2}
	(Sign)	(+)	(+)	(+)	(+)	(−)	(−)	(∼)

Table 4. *Cont.*

Algorithm		f_1	f_2	f_3	f_4	f_5	f_6	f_7
CLPSO	**Best**	3.40×10^{8}	$\mathbf{2.36 \times 10^{3}}$	1.44×10^{1}	9.67×10^{12}	1.45×10^{8}	1.64×10^{2}	4.03×10^{8}
	Median	4.20×10^{8}	$\mathbf{2.50 \times 10^{3}}$	1.47×10^{1}	1.44×10^{13}	1.89×10^{8}	$\mathbf{1.95 \times 10^{6}}$	6.80×10^{8}
	Worst	5.57×10^{8}	$\mathbf{2.76 \times 10^{3}}$	1.49×10^{1}	1.80×10^{13}	2.46×10^{8}	$\mathbf{3.24 \times 10^{6}}$	1.62×10^{9}
	Mean	4.24×10^{8}	$\mathbf{2.50 \times 10^{3}}$	1.47×10^{1}	1.38×10^{13}	1.99×10^{8}	$\mathbf{1.75 \times 10^{6}}$	7.10×10^{8}
	Std	5.95×10^{7}	$\mathbf{8.99 \times 10^{1}}$	1.41×10^{-1}	2.13×10^{12}	$\mathbf{3.06 \times 10^{7}}$	1.04×10^{6}	2.93×10^{8}
	***p*-value**	6.10×10^{-5}	6.10×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	$(+)$	$(-)$	$(+)$	$(+)$	$(-)$	$(-)$	$(+)$
CPSO-H$_6$	**Best**	1.16×10^{10}	1.27×10^{4}	2.04×10^{1}	1.18×10^{13}	3.98×10^{8}	1.88×10^{7}	2.09×10^{9}
	Median	1.80×10^{10}	1.37×10^{4}	2.05×10^{1}	4.14×10^{13}	4.99×10^{8}	1.94×10^{7}	9.69×10^{9}
	Worst	2.84×10^{10}	1.55×10^{4}	2.05×10^{1}	1.08×10^{14}	6.15×10^{8}	1.98×10^{7}	1.78×10^{10}
	Mean	1.83×10^{10}	1.39×10^{4}	2.05×10^{1}	4.19×10^{13}	4.94×10^{8}	1.94×10^{7}	8.70×10^{9}
	Std	5.47×10^{9}	8.98×10^{2}	$\mathbf{5.62 \times 10^{-2}}$	2.50×10^{13}	6.19×10^{7}	2.96×10^{5}	4.87×10^{9}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	7.30×10^{-2}	2.01×10^{-3}	6.10×10^{-5}
	(Sign)	$(+)$	$(+)$	$(+)$	$(+)$	$(\sim)$	$(-)$	$(+)$
HCBBPSO-JC	**Best**	7.01×10^{-20}	2.39×10^{3}	2.66×10^{0}	3.29×10^{11}	4.00×10^{8}	1.88×10^{7}	$\mathbf{1.39 \times 10^{4}}$
	Median	3.63×10^{-18}	2.72×10^{3}	$\mathbf{3.21 \times 10^{0}}$	$\mathbf{7.61 \times 10^{11}}$	5.57×10^{8}	1.98×10^{7}	2.37×10^{5}
	Worst	$\mathbf{1.01 \times 10^{-16}}$	2.92×10^{3}	$\mathbf{4.41 \times 10^{0}}$	$\mathbf{8.30 \times 10^{11}}$	7.33×10^{8}	1.99×10^{7}	2.69×10^{5}
	Mean	$\mathbf{1.57 \times 10^{-17}}$	2.65×10^{3}	$\mathbf{3.20 \times 10^{0}}$	$\mathbf{6.61 \times 10^{11}}$	5.66×10^{8}	1.97×10^{7}	2.08×10^{5}
	Std	$\mathbf{3.47 \times 10^{-17}}$	1.68×10^{2}	5.67×10^{-1}	$\mathbf{1.83 \times 10^{11}}$	1.01×10^{8}	2.61×10^{5}	7.51×10^{4}
	***p*-value**	9.34×10^{-1}	1.07×10^{-1}	4.54×10^{-1}	3.30×10^{-1}	5.24×10^{-1}	1.69×10^{-1}	2.15×10^{-2}
	(Sign)	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(+)$
HCBBPSO-JG	**Best**	$\mathbf{4.16 \times 10^{-20}}$	2.51×10^{3}	$\mathbf{2.08 \times 10^{0}}$	$\mathbf{2.93 \times 10^{11}}$	4.21×10^{8}	1.95×10^{7}	3.71×10^{4}
	Median	$\mathbf{7.76 \times 10^{-19}}$	2.80×10^{3}	3.40×10^{0}	8.84×10^{11}	5.58×10^{8}	1.98×10^{7}	$\mathbf{1.66 \times 10^{5}}$
	Worst	2.63×10^{-16}	3.04×10^{3}	5.18×10^{0}	1.38×10^{12}	7.30×10^{8}	1.99×10^{7}	$\mathbf{2.45 \times 10^{5}}$
	Mean	2.81×10^{-17}	2.77×10^{3}	3.47×10^{0}	7.82×10^{11}	5.54×10^{8}	1.97×10^{7}	$\mathbf{1.45 \times 10^{5}}$
	Std	6.88×10^{-17}	1.63×10^{2}	8.55×10^{-1}	3.23×10^{11}	9.93×10^{7}	$\mathbf{1.08 \times 10^{5}}$	$\mathbf{7.32 \times 10^{4}}$
Total # of $+/\sim/-$		5/1/0	4/1/1	5/1/0	5/1/0	0/2/4	0/2/4	4/2/0

Table 5. A comparison of the results of simulations performed by variants of the PSO algorithm. The best, median, worst, mean, standard deviation, *p*-value, and significance sign of the 25 runs when the NFEs counter reached 3.0×10^6 for the functions $f_8 \sim f_{14}$ are reported. The *p*-value was determined using the Wilcoxon signed-rank test between the best algorithm (HCBBPSO-JG) and the others. The significance sign "+" means that the HCBBPSO-JG algorithm was significantly better than the algorithm compared to it, the sign "∼" means that the two were not significantly different, and the sign "−" means that the HCBBPSO-JG algorithm was significantly worse than the algorithm compared to it. The measured value written in bold in colored cells for each function represents the value of the best algorithm for each statistical value.

Algorithm		f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
APSO	**Best**	2.51×10^8	3.20×10^{11}	2.39×10^4	2.18×10^2	4.33×10^5	3.06×10^{12}	3.11×10^{11}
	Median	1.07×10^9	3.76×10^{11}	2.43×10^4	2.27×10^2	5.67×10^5	3.43×10^{12}	4.24×10^{11}
	Worst	2.59×10^{10}	4.19×10^{11}	2.48×10^4	2.36×10^2	3.84×10^6	3.73×10^{12}	4.49×10^{11}
	Mean	3.70×10^9	3.73×10^{11}	2.43×10^4	2.27×10^2	8.47×10^5	3.43×10^{12}	4.13×10^{11}
	Std	6.47×10^9	2.56×10^{10}	3.21×10^2	6.02×10^0	8.83×10^5	1.56×10^{11}	3.67×10^{10}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	(+)	(+)	(+)	(+)	(+)	(+)	(+)
BBPSOjumpC	**Best**	6.65×10^6	3.80×10^8	$\mathbf{5.68 \times 10^3}$	1.54×10^2	3.98×10^5	3.97×10^4	1.54×10^9
	Median	7.00×10^7	6.40×10^8	6.49×10^3	1.67×10^2	4.82×10^5	9.33×10^4	2.13×10^9
	Worst	1.78×10^8	9.96×10^8	$\mathbf{6.78 \times 10^3}$	1.84×10^2	5.73×10^5	2.87×10^6	2.67×10^9
	Mean	7.05×10^7	6.17×10^8	$\mathbf{6.33 \times 10^3}$	1.68×10^2	4.74×10^5	3.39×10^5	2.15×10^9
	Std	4.81×10^7	1.62×10^8	3.47×10^2	8.39×10^0	5.36×10^4	7.22×10^5	2.95×10^8
	***p*-value**	9.46×10^{-2}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	(∼)	(+)	(−)	(−)	(+)	(+)	(+)
BBPSOjumpG	**Best**	2.91×10^7	4.07×10^8	5.75×10^3	1.54×10^2	4.02×10^5	3.73×10^4	1.59×10^9
	Median	8.60×10^7	5.41×10^8	$\mathbf{6.31 \times 10^3}$	1.73×10^2	4.74×10^5	6.59×10^4	2.03×10^9
	Worst	1.68×10^8	1.01×10^9	6.93×10^3	1.83×10^2	6.44×10^5	2.19×10^5	2.84×10^9
	Mean	7.58×10^7	5.75×10^8	6.39×10^3	1.71×10^2	4.89×10^5	8.88×10^4	2.08×10^9
	Std	4.23×10^7	1.47×10^8	4.19×10^2	9.20×10^0	7.62×10^4	5.95×10^4	3.55×10^8
	***p*-value**	3.53×10^{-2}	6.10×10^{-5}	1.83×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	(+)	(+)	(−)	(−)	(+)	(+)	(+)

Table 5. *Cont.*

Algorithm		f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
CLPSO	**Best**	6.93×10^{6}	1.31×10^{9}	9.05×10^{3}	$\mathbf{8.89 \times 10^{1}}$	5.57×10^{5}	8.02×10^{8}	2.17×10^{9}
	Median	1.60×10^{7}	1.42×10^{9}	9.30×10^{3}	$\mathbf{1.10 \times 10^{2}}$	5.96×10^{5}	1.02×10^{9}	2.43×10^{9}
	Worst	$\mathbf{1.34 \times 10^{8}}$	1.86×10^{9}	9.67×10^{3}	$\mathbf{1.29 \times 10^{2}}$	6.55×10^{5}	1.78×10^{9}	2.72×10^{9}
	Mean	$\mathbf{3.53 \times 10^{7}}$	1.48×10^{9}	9.32×10^{3}	$\mathbf{1.09 \times 10^{2}}$	6.01×10^{5}	1.10×10^{9}	2.45×10^{9}
	Std	$\mathbf{3.85 \times 10^{7}}$	1.51×10^{8}	$\mathbf{1.58 \times 10^{2}}$	1.23×10^{1}	2.76×10^{4}	2.97×10^{8}	1.49×10^{8}
	***p*-value**	7.62×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	$(\sim)$	$(+)$	$(+)$	$(-)$	$(+)$	$(+)$	$(+)$
CPSO-H$_6$	**Best**	1.27×10^{9}	1.24×10^{10}	1.35×10^{4}	2.22×10^{2}	2.49×10^{6}	7.71×10^{10}	1.16×10^{10}
	Median	3.46×10^{9}	1.90×10^{10}	1.45×10^{4}	2.24×10^{2}	2.90×10^{6}	1.30×10^{11}	1.42×10^{10}
	Worst	2.26×10^{10}	2.78×10^{10}	1.58×10^{4}	2.26×10^{2}	3.24×10^{6}	3.02×10^{11}	2.08×10^{10}
	Mean	5.23×10^{9}	1.88×10^{10}	1.45×10^{4}	2.24×10^{2}	2.88×10^{6}	1.55×10^{11}	1.54×10^{10}
	Std	5.27×10^{9}	3.87×10^{9}	4.92×10^{2}	1.13×10^{0}	1.97×10^{5}	6.96×10^{10}	2.74×10^{9}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	$(+)$	$(+)$	$(+)$	$(+)$	$(+)$	$(+)$	$(+)$
HCBBPSO-JC	**Best**	$\mathbf{1.06 \times 10^{4}}$	2.51×10^{7}	6.49×10^{3}	2.17×10^{2}	2.36×10^{3}	9.14×10^{2}	$\mathbf{7.99 \times 10^{7}}$
	Median	$\mathbf{7.04 \times 10^{6}}$	$\mathbf{2.96 \times 10^{7}}$	7.15×10^{3}	2.18×10^{2}	3.01×10^{3}	$\mathbf{1.41 \times 10^{3}}$	9.72×10^{7}
	Worst	1.68×10^{8}	$\mathbf{3.20 \times 10^{7}}$	7.62×10^{3}	2.18×10^{2}	3.91×10^{3}	$\mathbf{3.35 \times 10^{3}}$	$\mathbf{1.09 \times 10^{8}}$
	Mean	5.58×10^{7}	$\mathbf{2.96 \times 10^{7}}$	7.15×10^{3}	2.18×10^{2}	3.00×10^{3}	$\mathbf{1.52 \times 10^{3}}$	$\mathbf{9.54 \times 10^{7}}$
	Std	7.00×10^{7}	$\mathbf{1.84 \times 10^{6}}$	3.37×10^{2}	$\mathbf{2.66 \times 10^{-1}}$	4.59×10^{2}	$\mathbf{6.12 \times 10^{2}}$	8.54×10^{6}
	***p*-value**	4.21×10^{-1}	5.42×10^{-1}	8.04×10^{-1}	8.04×10^{-1}	9.46×10^{-2}	8.90×10^{-1}	6.39×10^{-1}
	(Sign)	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$	$(\sim)$
HCBBPSO-JG	**Best**	8.08×10^{4}	$\mathbf{2.08 \times 10^{7}}$	6.59×10^{3}	2.17×10^{2}	$\mathbf{2.24 \times 10^{3}}$	$\mathbf{6.96 \times 10^{2}}$	8.66×10^{7}
	Median	1.68×10^{7}	2.97×10^{7}	7.23×10^{3}	2.18×10^{2}	$\mathbf{2.69 \times 10^{3}}$	1.41×10^{3}	$\mathbf{9.60 \times 10^{7}}$
	Worst	1.58×10^{8}	3.67×10^{7}	7.79×10^{3}	2.18×10^{2}	$\mathbf{3.30 \times 10^{3}}$	3.35×10^{3}	1.10×10^{8}
	Mean	3.97×10^{7}	3.00×10^{7}	7.21×10^{3}	2.18×10^{2}	$\mathbf{2.74 \times 10^{3}}$	1.58×10^{3}	9.65×10^{7}
	Std	5.30×10^{7}	4.15×10^{6}	3.53×10^{2}	3.42×10^{-1}	$\mathbf{2.82 \times 10^{2}}$	7.92×10^{2}	$\mathbf{6.88 \times 10^{6}}$
Total # of $+/\sim/-$		3/3/0	5/1/0	3/1/2	2/1/3	5/1/0	5/1/0	5/1/0

Table 6. A comparison of the results of simulations performed by variants of the PSO algorithm. The best, median, worst, mean, standard deviation, *p*-value, and significance sign of the 25 runs when the NFEs counter reached 3.0×10^6 for the functions $f_{15} \sim f_{20}$ are reported. The *p*-value was determined using the Wilcoxon signed-rank test between the best algorithm (HCBBPSO-JG) and the others. The significance sign "+" means that the HCBBPSO-JG algorithm was significantly better than the algorithm compared to it, the sign "∼" means that the two were not significantly different, and the sign "−" means that the HCBBPSO-JG algorithm was significantly worse than the algorithm compared to it. The measured value written in bold in colored cells for each function represents the value of the best algorithm for each statistical value.

Algorithm		f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
APSO	**Best**	2.39×10^4	4.22×10^2	1.64×10^6	7.02×10^{12}	5.93×10^6	7.91×10^{12}
	Median	2.42×10^4	4.28×10^2	5.13×10^6	7.60×10^{12}	7.91×10^6	8.37×10^{12}
	Worst	2.50×10^4	4.30×10^2	5.38×10^7	7.99×10^{12}	2.81×10^7	8.54×10^{12}
	Mean	2.43×10^4	4.28×10^2	9.75×10^6	7.55×10^{12}	1.10×10^7	8.34×10^{12}
	Std	3.21×10^2	1.96×10^0	1.45×10^7	2.57×10^{11}	5.96×10^6	1.61×10^{11}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^5	6.10×10^{-5}
	(Sign)	(+)	(+)	(+)	(+)	(+)	(+)
BBPSOjumpC	**Best**	7.39×10^3	3.45×10^2	1.21×10^6	4.33×10^8	3.10×10^6	1.46×10^8
	Median	7.95×10^3	3.62×10^2	1.56×10^6	2.39×10^9	3.70×10^6	9.56×10^8
	Worst	8.92×10^3	3.77×10^2	2.05×10^6	1.07×10^{10}	4.02×10^6	1.49×10^{10}
	Mean	8.07×10^3	3.62×10^2	1.58×10^6	3.92×10^9	3.72×10^6	2.69×10^9
	Std	5.43×10^2	8.43×10^0	2.30×10^5	3.26×10^9	2.72×10^5	3.84×10^9
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	(−)	(−)	(+)	(+)	(+)	(+)
BBPSOjumpG	**Best**	$\mathbf{6.78 \times 10^3}$	3.50×10^2	1.27×10^6	1.46×10^9	3.34×10^6	9.52×10^8
	Median	$\mathbf{7.89 \times 10^3}$	3.67×10^2	1.56×10^6	3.29×10^9	3.66×10^6	2.10×10^9
	Worst	$\mathbf{8.85 \times 10^3}$	3.89×10^2	1.86×10^6	1.85×10^{10}	4.30×10^6	3.87×10^{10}
	Mean	$\mathbf{7.85 \times 10^3}$	3.68×10^2	1.58×10^6	4.29×10^9	3.70×10^6	7.95×10^9
	Std	5.34×10^2	1.19×10^1	1.74×10^5	4.13×10^9	2.27×10^5	1.13×10^{10}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	(−)	(−)	(+)	(+)	(+)	(+)

Table 6. *Cont.*

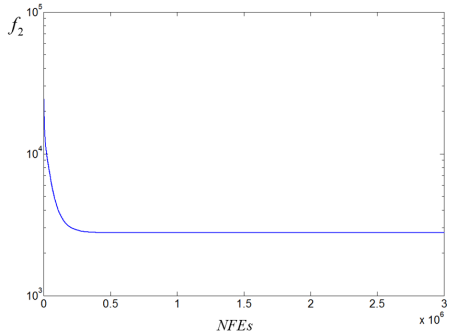
Algorithm		f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
CLPSO	**Best**	1.36×10^{4}	$\mathbf{2.64 \times 10^{2}}$	1.20×10^{6}	3.32×10^{10}	5.39×10^{6}	4.55×10^{10}
	Median	1.37×10^{4}	$\mathbf{2.98 \times 10^{2}}$	1.41×10^{6}	3.92×10^{10}	5.85×10^{6}	5.61×10^{10}
	Worst	1.39×10^{4}	$\mathbf{3.10 \times 10^{2}}$	1.55×10^{6}	4.54×10^{10}	6.54×10^{6}	6.16×10^{10}
	Mean	1.37×10^{4}	$\mathbf{2.95 \times 10^{2}}$	1.39×10^{6}	3.90×10^{10}	5.86×10^{6}	5.54×10^{10}
	Std	$\mathbf{9.77 \times 10^{1}}$	1.28×10^{1}	9.71×10^{4}	4.14×10^{9}	3.12×10^{5}	4.51×10^{9}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	$(+)$	$(-)$	$(+)$	$(+)$	$(+)$	$(+)$
CPSO-H$_6$	**Best**	1.34×10^{4}	4.06×10^{2}	2.86×10^{6}	1.52×10^{12}	1.00×10^{7}	1.58×10^{12}
	Median	1.41×10^{4}	4.08×10^{2}	3.20×10^{6}	1.78×10^{12}	1.33×10^{7}	1.94×10^{12}
	Worst	1.49×10^{4}	4.12×10^{2}	3.77×10^{6}	2.12×10^{12}	2.30×10^{7}	2.35×10^{12}
	Mean	1.41×10^{4}	4.09×10^{2}	3.23×10^{6}	1.76×10^{12}	1.40×10^{7}	1.98×10^{12}
	Std	4.21×10^{2}	1.89×10^{0}	2.70×10^{5}	1.56×10^{11}	4.03×10^{6}	1.97×10^{11}
	***p*-value**	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
	(Sign)	$(+)$	$(+)$	$(+)$	$(+)$	$(+)$	$(+)$
HCBBPSO-JC	**Best**	1.07×10^{4}	3.96×10^{2}	1.59×10^{4}	2.24×10^{3}	$\mathbf{4.50 \times 10^{5}}$	1.49×10^{3}
	Median	1.16×10^{4}	3.97×10^{2}	2.10×10^{4}	3.91×10^{3}	7.92×10^{5}	$\mathbf{1.69 \times 10^{3}}$
	Worst	1.20×10^{4}	3.97×10^{2}	2.38×10^{4}	$\mathbf{5.65 \times 10^{3}}$	$\mathbf{9.25 \times 10^{5}}$	2.07×10^{3}
	Mean	1.15×10^{4}	3.97×10^{2}	2.07×10^{4}	4.00×10^{3}	7.71×10^{5}	1.74×10^{3}
	Std	4.30×10^{2}	3.97×10^{-1}	2.37×10^{3}	$\mathbf{8.94 \times 10^{2}}$	$\mathbf{1.59 \times 10^{5}}$	1.64×10^{2}
	***p*-value**	7.20×10^{-1}	3.02×10^{-2}	4.27×10^{-4}	4.46×10^{-1}	1.00×10^{0}	1.88×10^{-1}
	(Sign)	$(\sim)$	$(+)$	$(+)$	$(\sim)$	$(\sim)$	$(\sim)$
HCBBPSO-JG	**Best**	1.06×10^{4}	3.96×10^{2}	$\mathbf{1.37 \times 10^{4}}$	$\mathbf{1.90 \times 10^{3}}$	5.29×10^{5}	$\mathbf{1.40 \times 10^{3}}$
	Median	1.16×10^{4}	3.96×10^{2}	$\mathbf{1.77 \times 10^{4}}$	$\mathbf{3.69 \times 10^{3}}$	$\mathbf{7.14 \times 10^{5}}$	1.70×10^{3}
	Worst	1.23×10^{4}	3.97×10^{2}	$\mathbf{2.12 \times 10^{4}}$	6.46×10^{3}	1.06×10^{6}	$\mathbf{1.92 \times 10^{3}}$
	Mean	1.15×10^{4}	3.96×10^{2}	$\mathbf{1.77 \times 10^{4}}$	$\mathbf{3.61 \times 10^{3}}$	$\mathbf{7.60 \times 10^{5}}$	$\mathbf{1.69 \times 10^{3}}$
	Std	4.93×10^{2}	$\mathbf{3.87 \times 10^{-1}}$	$\mathbf{1.97 \times 10^{3}}$	1.28×10^{3}	1.62×10^{5}	$\mathbf{1.57 \times 10^{2}}$
Total # of $+/\sim/-$		3/2/1	3/0/3	6/0/0	5/1/0	5/1/0	5/1/0

4.3. The Results of Simulations Using the HCBBPSO-JG Algorithm for the CEC 2010 Benchmark Functions

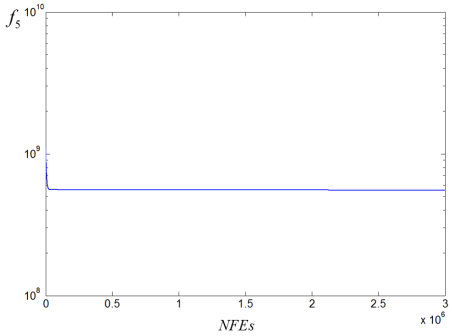
Table 7 include the simulation results of the best algorithm, the HCBBPSO-JG algorithm. The best, median, worst, mean, and standard deviation of the 25 runs when the NFEs counter reached 1.2×10^5, 6.0×10^5, and 3.0×10^6 are reported. This information ws used in the CEC 2010 LSGO Challenge scoring system, and the CEC 2010 LSGO Challenge included a rule stating that this information must be opened with the single convergence curves for several functions that were specified in advance. Figure 3 show the convergence curves for the designated functions, f_2, f_5, f_8, f_{10}, f_{13}, f_{15}, f_{18}, and f_{20}. For each function, the curve has been graphed using the average of the results of all 25 runs. As mentioned previously, the HCBBPSO-JG algorithm converged prematurely for the functions f_2, f_5, f_{10}, and f_{15}, which were related to Rastrigin's function. For the other functions, there were improvements until the maximum NFEs was reached, which helped the HCBBPSO-JG algorithm perform well.

Table 7. The results of simulations using the best algorithm, the HCBBPSO-JG algorithm: the best, median, worst, mean, and standard deviation of the 25 runs when the NFEs counter reached 1.2×10^5, 6.0×10^5, and 3.0×10^6 are reported.

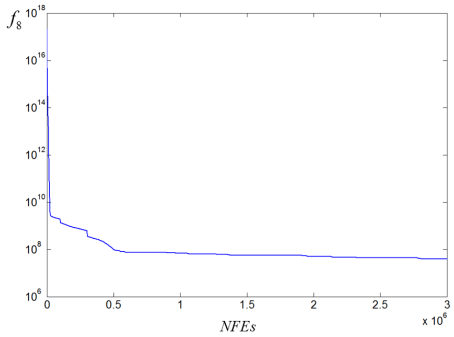
1000D		f_1	f_2	f_3	f_4	f_5	f_6	f_7
1.2×10^5	**Best**	7.15×10^8	3.47×10^3	7.28×10^0	7.35×10^{12}	4.21×10^8	1.95×10^7	7.55×10^9
	Median	1.36×10^9	3.84×10^3	8.34×10^0	1.78×10^{13}	5.58×10^8	1.98×10^7	1.06×10^{10}
	Worst	2.35×10^9	3.98×10^3	9.12×10^0	2.17×10^{13}	7.34×10^8	1.99×10^7	1.76×10^{10}
	Mean	1.33×10^9	3.79×10^3	8.26×10^0	1.58×10^{13}	5.57×10^8	1.97×10^7	1.11×10^{10}
	Std	3.87×10^8	1.48×10^2	5.62×10^{-1}	5.12×10^{12}	1.04×10^8	9.64×10^4	2.84×10^9
6.0×10^5	**Best**	5.69×10^1	2.51×10^3	2.08×10^0	1.32×10^{12}	4.21×10^8	1.95×10^7	1.38×10^8
	Median	2.03×10^2	2.80×10^3	3.40×10^0	3.61×10^{12}	5.58×10^8	1.98×10^7	2.89×10^8
	Worst	3.94×10^3	3.04×10^3	5.18×10^0	5.66×10^{12}	7.34×10^8	1.99×10^7	5.39×10^8
	Mean	4.44×10^2	2.78×10^3	3.47×10^0	3.67×10^{12}	5.57×10^8	1.97×10^7	3.00×10^8
	Std	9.77×10^2	1.62×10^2	8.55×10^{-1}	1.38×10^{12}	1.04×10^8	9.79×10^4	1.11×10^8
3.0×10^6	**Best**	4.16×10^{-20}	2.51×10^3	2.08×10^0	2.93×10^{11}	4.21×10^8	1.95×10^7	3.71×10^4
	Median	7.76×10^{-19}	2.80×10^3	3.40×10^0	8.84×10^{11}	5.58×10^8	1.98×10^7	1.66×10^5
	Worst	2.63×10^{-16}	3.04×10^3	5.18×10^0	1.38×10^{12}	7.30×10^8	1.99×10^7	2.45×10^5
	Mean	2.81×10^{-17}	2.77×10^3	3.47×10^0	7.82×10^{11}	5.54×10^8	1.97×10^7	1.45×10^5
	Std	6.88×10^{-17}	1.63×10^2	8.55×10^{-1}	3.23×10^{11}	9.93×10^7	1.08×10^5	7.32×10^4
1000D		f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
1.2×10^5	**Best**	6.32×10^7	9.32×10^8	7.64×10^3	2.18×10^2	7.09×10^5	2.04×10^6	2.40×10^9
	Median	1.63×10^8	1.10×10^9	8.38×10^3	2.19×10^2	8.67×10^5	3.96×10^6	2.75×10^9
	Worst	7.10×10^9	1.60×10^9	8.85×10^3	2.20×10^2	9.79×10^5	7.47×10^6	3.09×10^9
	Mean	1.24×10^9	1.13×10^9	8.25×10^3	2.19×10^2	8.80×10^5	4.05×10^6	2.76×10^9
	Std	2.38×10^9	1.51×10^8	3.42×10^2	3.85×10^{-1}	7.55×10^4	1.77×10^6	2.31×10^8
6.0×10^5	**Best**	1.29×10^6	1.24×10^8	6.59×10^3	2.17×10^2	1.50×10^5	1.67×10^3	4.86×10^8
	Median	6.85×10^7	1.65×10^8	7.27×10^3	2.18×10^2	1.72×10^5	5.39×10^3	5.37×10^8
	Worst	2.26×10^8	1.98×10^8	7.79×10^3	2.18×10^2	1.96×10^5	1.54×10^4	6.23×10^8
	Mean	7.48×10^7	1.64×10^8	7.21×10^3	2.18×10^2	1.72×10^5	5.76×10^3	5.41×10^8
	Std	6.83×10^7	1.82×10^7	3.53×10^2	3.41×10^{-1}	1.44×10^4	4.08×10^3	4.43×10^7
3.0×10^6	**Best**	8.08×10^4	2.08×10^7	6.59×10^3	2.17×10^2	2.24×10^3	6.96×10^2	8.66×10^7
	Median	1.68×10^7	2.97×10^7	7.23×10^3	2.18×10^2	2.69×10^3	1.41×10^3	9.60×10^7
	Worst	1.58×10^8	3.67×10^7	7.79×10^3	2.18×10^2	3.30×10^3	3.35×10^3	1.10×10^8
	Mean	3.97×10^7	3.00×10^7	7.21×10^3	2.18×10^2	2.74×10^3	1.58×10^3	9.65×10^7
	Std	5.30×10^7	4.15×10^6	3.53×10^2	3.42×10^{-1}	2.82×10^2	7.92×10^2	6.88×10^6
1000D		f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	
1.2×10^5	**Best**	1.11×10^4	4.00×10^2	1.65×10^6	8.22×10^8	5.37×10^6	8.05×10^8	
	Median	1.21×10^4	4.01×10^2	1.82×10^6	1.12×10^9	7.16×10^6	1.05×10^9	
	Worst	1.28×10^4	4.02×10^2	2.03×10^6	1.73×10^9	8.44×10^6	1.76×10^9	
	Mean	1.20×10^4	4.01×10^2	1.84×10^6	1.13×10^9	7.02×10^6	1.14×10^9	
	Std	5.03×10^2	7.48×10^{-1}	1.16×10^5	2.86×10^8	8.69×10^5	2.39×10^8	
6.0×10^5	**Best**	1.06×10^4	3.96×10^2	4.02×10^5	2.13×10^4	2.40×10^6	3.14×10^3	
	Median	1.16×10^4	3.96×10^2	4.69×10^5	3.98×10^4	2.79×10^6	3.55×10^3	
	Worst	1.23×10^4	3.97×10^2	5.25×10^5	6.12×10^4	3.55×10^6	4.52×10^3	
	Mean	1.15×10^4	3.96×10^2	4.66×10^5	4.10×10^4	2.86×10^6	3.63×10^3	
	Std	4.93×10^2	3.87×10^{-1}	3.38×10^4	1.17×10^4	3.39×10^5	3.71×10^2	
3.0×10^6	**Best**	1.06×10^4	3.96×10^2	1.37×10^4	1.90×10^3	5.29×10^5	1.40×10^3	
	Median	1.16×10^4	3.96×10^2	1.77×10^4	3.69×10^3	7.14×10^5	1.70×10^3	
	Worst	1.23×10^4	3.97×10^2	2.12×10^4	6.46×10^3	1.06×10^6	1.92×10^3	
	Mean	1.15×10^4	3.96×10^2	1.77×10^4	3.61×10^3	7.60×10^5	1.69×10^3	
	Std	4.93×10^2	3.87×10^{-1}	1.97×10^3	1.28×10^3	1.62×10^5	1.57×10^2	



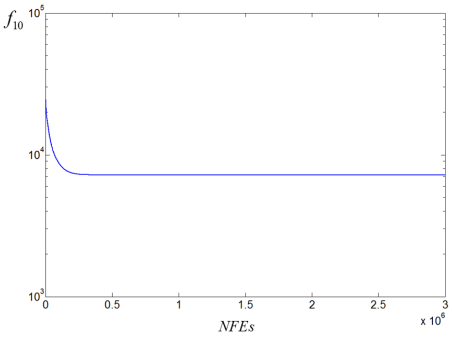
(a) f_2: Shifted Rastrigin's Function



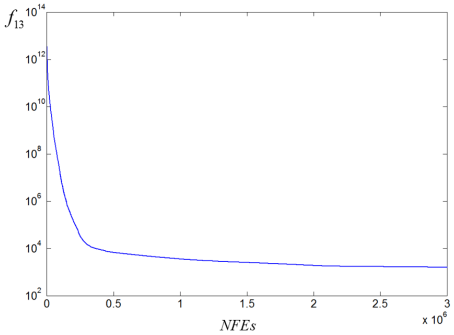
(b) f_5: Single-group Shifted m-roatated Rastrigin's Function



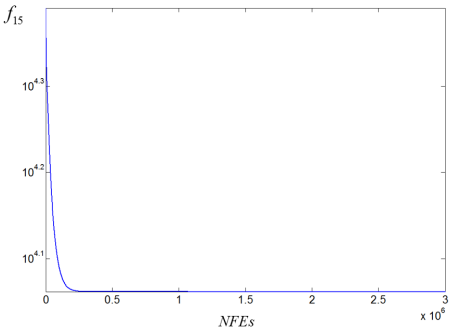
(c) f_8: Single-group Shifted m-dimensional Rosenbrock's Function



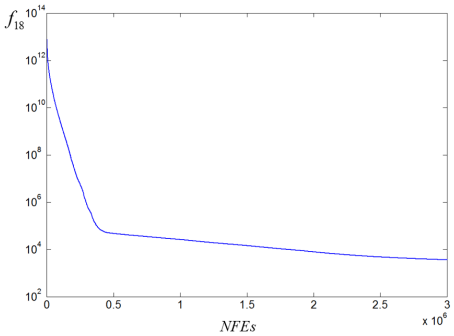
(d) f_{10}: $\frac{D}{2m}$-group Shifted and m-rotated Rastrigin's Function



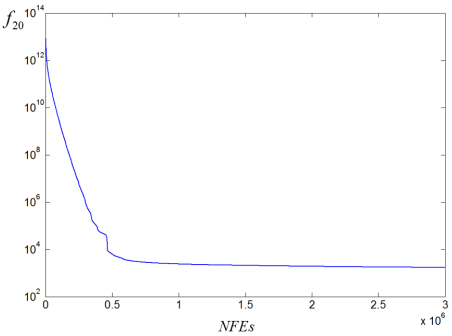
(e) f_{13}: $\frac{D}{2m}$-group Shifted m-dimensional Rosenbrock's Function



(f) f_{15}: $\frac{D}{m}$-group Shifted and m-rotated Rastrigin's Function



(g) f_{18}: $\frac{D}{m}$-group Shifted m-dimensional Rosenbrock's Function



(h) f_{20}: Shifted Rosenbrock's Function

Figure 3. Single convergence curves of HCBBPSO-JG for the functions f_2, f_5, f_8, f_{10}, f_{13}, f_{15}, f_{18}, and f_{20}: The curves have been graphed using the average of the results of all 25 runs.

5. Conclusions

This paper proposed heterogeneous cooperative BBPSO algorithms that used a jumping strategy, which was strengthened for use with high-dimensional optimization problems by combining an improved exploration ability, which was introduced by means of heterogeneous cooperation and the jumping strategy, with the merits of BBPSO, which is simple but robust because it does not need to consider the selection of controllable parameters for the PSO algorithm and because its performance is not affected by the values of these parameters.

In the comparative simulations based on the 20 qualified benchmark functions and the evaluation system used in the CEC 2010 LSGO Challenge, the HCBBPSO-Jx algorithms provided improved results for most of the functions; notably, the HCBBPSO-JG algorithm performed the best according to the overall evaluation criteria. Although the proposed algorithms converged prematurely for several benchmark functions that were related to Rastrigin's function, they will be improved in future work. Therefore, the results of this study lead us to conclude that the proposed HCBBPSO-JG algorithm is useful as an optimizer for solving high-dimensional problems. In future work, the proposed algorithm will be improved for Rastrigin's function, tested with additional benchmark functions and compared with other state-of-the-art optimizers. We will also study the parameter split factor K, because it is very hard to select K depending on the problem. In particular, the problems to solve with the algorithms proposed in this paper are high-dimensional problems with high complexity. In addition, the further study will also be conducted on other parameters to improve the performance of the proposed algorithm.

Author Contributions: Conceptualization, J.L. and W.K.; methodology, J.L.; software, J.L. and W.K.; validation, J.L. and W.K.; formal analysis, J.L.; investigation, J.L.; resources, J.L.; data curation, J.L.; writing—original draft preparation, J.L. and W.K.; writing—review and editing, J.L. and W.K.; visualization, J.L.; supervision, J.L. and W.K.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1C1C1008520).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lei, G.; Shao, K.R.; Guo, Y.; Zhu, J.; Lavers, J.D. Improved Sequential Optimization Method for High Dimensional Electromagnetic Device Optimization. *IEEE Trans. Magn.* **2009**, *45*, 3993–3996.
2. Xue, Y.; Tang, T.; Liu, A.X. Large-Scale Feedforward Neural Network Optimization by a Self-Adaptive Strategy and Parameter Based Particle Swarm Optimization. *IEEE Access* **2019**, *7*, 52473–52483. [CrossRef]
3. Qiao, W.; Yang, Z. Modified Dolphin Swarm Algorithm Based on Chaotic Maps for Solving High-Dimensional Function Optimization Problems. *IEEE Access* **2019**, *7*, 110472–110486. [CrossRef]
4. Gardeux, V.; Chelouah, R.; Siarry, P.; Glover, F. EM323: A line search based algorithm for solving high-dimensional continuous non-linear optimization problems. *Soft Comput.* **2011**, *15*, 2275–2285. [CrossRef]
5. Shan, S.; Wang, G.G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Struct. Multidisc. Optim.* **2010**, *41*, 219–241. [CrossRef]
6. Wang, H.; Rahnamayan, S.; Wu, Z. Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. *J. Parallel Distrib. Comput.* **2013**, *73*, 62–73. [CrossRef]
7. Wang, S.; Liu, G.; Gao, M.; Cao, S.; Guo, A.; Wang, J. Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators. *Inf. Sci.* **2020**, *540*, 175–201. [CrossRef]
8. Borowska, B. Genetic Learning Particle Swarm Optimization with Interlaced Ring Topology. In *International Conference on Computational Science*; Lecture Notes in Computer Science; Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J., Eds.; Springer: Cham, Switzerland, 2020; Volume 12141, pp. 136–148.

9. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
10. Shi, Y.; Eberhart, R. A Modified Particle Swarm Optimizer. In Proceedings of the IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
11. Clerc, M.; Kennecy, J. The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
12. Shi, Y.; Eberhart, R.C. Empirical Study of Particle Swarm Optimization. In Proceedings of the IEEE Cngress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; pp. 1945–1950.
13. Shi, Y.; Eberhart, R.C. Parameter Selection in Particle Swarm Optimization. *Evol. Program. VII* **1998**, *1447*, 591–600.
14. Trelea, I.C. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Inf. Process. Lett.* **2003**, *85*, 317–325. [CrossRef]
15. Coelho, L.S.; Krohling, R.A. Predictive Controller Tuning Using Modified Particle Swarm Optimization Based on Cauchy and Gaussian Distributions. *Adv. Soft Comput.* **2005**, *32*, 287–298.
16. Krohling, R.A.; Hoffmann, F.; Coelho, L.S. Co-evolutionary Particle Swarm Optimization for Min-Max Problems using Gaussian Distribution. In Proceedings of the IEEE Cngress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; pp. 959–964.
17. Krohling, R.A.; Mendel, E. Bare Bones Particle Swarm Optimization with Gaussian or Cauchy Jumps. In Proceedings of the IEEE Cngress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 3285–3291.
18. Lee, J.-W.; Lee, J.-J. Gaussian-Distributed Particle Swarm Optimization: A Novel Gaussian Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Industrial Technology, Cape Town, South Africa, 25–28 February 2013; pp. 1122–1127.
19. Kennedy, J. Bare bones particle swarms. In Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, 26 April 2003; pp. 80–87.
20. Van den Bergh, F.; Engelbrecht, A.P. A Cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
21. Krohling, R.A. Gaussian particle swarm with jumps. In Proceedings of the IEEE Cngress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1226–1231.
22. Blackwell, T. A Study of Collapse in Bare Bones Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2012**, *16*, 354–372. [CrossRef]
23. Al-Rifaie, M.M.; Blackwell, T. Cognitive Bare Bones Particle Swarm Optimisation with Jumps. *Int. J. Swarm Intell. Res.* **2016**, *7*, 1–31. [CrossRef]
24. Tang, K.; Li, X.; Suganthan, P.N.; Yang, Z.; Weise, T. *Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization*; Technical Report; Nat. Inspir. Comput. Appl. Lab., USTC: Hefei, China, 2009. Available online: http://nical.ustc.edu.cn/cec10ss.php (accessed on 17 October 2011).
25. Zhan, Z.-H.; Zhang, J.; Li, Y.; Chung, H.S.-H. Adaptive Particle Swarm Optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2009**, *39*, 1362–1381. [CrossRef] [PubMed]
26. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]