electronics

MDPI

Article

Particle Swarm Optimization Combined with Inertia-Free Velocity and Direction Search

Kun Miao *, Qian Feng and Wei Kuang

School of Civil Engineering, Central South University, Changsha 410075, China; 184811138@csu.edu.cn (Q.F.); kiwi21@csu.edu.cn (W.K.)

* Correspondence: miao98004@csu.edu.cn

Abstract: The particle swarm optimization algorithm (PSO) is a widely used swarm-based natural inspired optimization algorithm. However, it suffers search stagnation from being trapped into a sub-optimal solution in an optimization problem. This paper proposes a novel hybrid algorithm (SDPSO) to improve its performance on local searches. The algorithm merges two strategies, the static exploitation (SE, a velocity updating strategy considering inertia-free velocity), and the direction search (DS) of Rosenbrock method, into the original PSO. With this hybrid, on the one hand, extensive exploration is still maintained by PSO; on the other hand, the SE is responsible for locating a small region, and then the DS further intensifies the search. The SDPSO algorithm was implemented and tested on unconstrained benchmark problems (CEC2014) and some constrained engineering design problems. The performance of SDPSO is compared with that of other optimization algorithms, and the results show that SDPSO has a competitive performance.

Keywords: particle swarm optimization; direction search method; Rosenbrock method; global optimization

Citation: Miao, K.; Feng, Q.; Kuang, W. Particle Swarm Optimization Combined with Inertia-Free Velocity and Direction Search. *Electronics* **2021**, *10*, 597. https://doi.org/10.3390/electronics10050597

Academic Editor: Javid Taheri

Received: 29 January 2021
Accepted: 27 February 2021
Published: 4 March 2021

1. Introduction

Optimization problems are difficult to be solved and commonly arise in fields such as engineering [1], management [2], aerospace technology [3] and scientific research [4]. Such optimization problems are often multidimensional, complex, and time consuming. Solving these problems has always attracted extensive research interest.

The meta-heuristic algorithms, due to their simplicity and flexibility, have been widely used. In the past few decades, they have achieved great development [5]. The PSO algorithm is a meta-heuristic algorithm that simulates the natural tendency of birds to find food, and it is also one of the most popular meta-heuristic technologies for large solution spaces with large numbers of peaks. Aler et al. [6] has made a comprehensive review for the PSO algorithm in detail.

Although the PSO algorithm presents fast convergence [7] with a good exploration ability, it is susceptible to premature over the complex fitness landscapes [8]. The important reason of affecting the convergence is that the PSO is strong randomness, making optimization process degenerate to half-blind state, thereby lead to the poor local search ability and slowing convergence rate.

The randomness is influenced by the velocity of particles in PSO; however, the velocity is difficult to be adjusted or controlled through algorithm parameters. Although there are many methods with adaptive parameters adjustment to the velocity [9], the adjustment does not depend on the positions but only the iterative process. Thus, some areas near where particles go may be neglected during the search process for too fast or too slow velocity. Considering that particles are allowed to forget their own flying experience, we let their history velocity no longer work to weak the randomness for locating possible local areas. That is to say, we set the inertia weight in velocity formula to be zero and call the search with the inertia-free velocity the ***static exploitation*** (**SE**).

Further, we want to exploit the local areas for the best solution. Rosenbrock method [10], which is a form of direct search, has the advantages of fast convergence, and it often not only identifies a ridge but also performs better on functions with sharp ridges [11]. In fact, the best solutions are often just located on ridges for complex functions, and a successful search on such ridges is usually the key to solving most complex problems. Thus, the ***direction search*** (DS), a component in the Rosenbrock method, is introduced into our algorithm for a fine search.

With the above considerations, this paper presents an improved PSO algorithm with an inertia-free velocity and direction search (SDPSO), which combines the SE and the DS with the original PSO. The SE means that a particle moves at lower velocity to balance the particle velocity; and the DS means that the search is strengthened in local areas.

The paper is organized as follows: the next section, Section 2, describes some research on PSO variants. The SE with inertia-free velocity from the original PSO is described in Section 3. The DS based on the Rosenbrock method is described in detail in Section 4. With Sections 3 and 4, the SDPSO algorithm procedure is described in Section 5. In Section 6, unconstrained benchmark problems (CEC2014) and some constrained engineering design problems are tested and compared with some advanced algorithms to demonstrate the performance of the SDPSO. In Section 7, parameter sensitivity analyses for SDPSO were conducted. Finally, conclusions are given in Section 8.

2. Review of Improving the PSO Algorithm

Many studies have attempted to improve the classical PSO algorithm in a variety of strategies. Frans et al. [12] proposed the cooperative evolutionary framework using multiple swarms to optimize different components of the solution vector cooperatively. Liang et al. [13] introduced a comprehensive learning PSO that can scan larger search spaces and increase the probability of finding global optimality. Liang et al. [14] divided the entire population into small swarms and regrouped them with information exchange. Sun, L. et al. [15] presented a cooperative particle swarm optimizer with statistical variable interdependence learning. Li et al. [16] generated candidate positions by estimating the distribution information of historical promising individual best position with an estimation of distribution algorithm. Gülcü et al. [17] took the master-slave paradigm for multiple groups to set up a comprehensive learning particle swarm optimizer. Considering multi-population cooperation, Li et al. [18] constructed learning exemplars for information sharing with a multidimensional comprehensive learning strategy and a dynamic segment-based mean learning strategy. Xu et al. [19] proposed a dimensional learning strategy, in which the personal best position of a particle learns from the global best position in a way of dimension by dimension to construct the learning exemplar. Considering neighborhood and historical memory, Li [20] proposed that every particle' position is guided by the experience of its neighbors and its competitors for decreasing the risk of premature convergence.

However, velocity update and mixing with other algorithms are the main strategies for PSO in this paper. Hence, these two aspects are reviewed below.

2.1. Modifying the Velocity Update Strategy of Particles

He et al. [21] introduced the biological force to the velocity update equation for preserving swarm integrity. Considering that the velocity of a bird is decided by both its current velocity and the acceleration, Zeng [22] introduced the acceleration to the PSO algorithm. To avoid premature convergence in the early process, Chen [23] set a function of sine cosine as acceleration coefficients of the velocity update equation. Liu et al. [24] thought frequent velocity update is not good for local exploitation of particles, so the velocities of the particles should be not always updated in a continuous way. To restrict the particles in the feasible range, Liu et al. [25] introduced the momentum factor into the position update equation. Modifying the limit of the speeds or the positions of the particles may also improve performance of PSO algorithm. Stacey et al. [26] proposed a speed limit

for re-randomization speed and a position limit for re-randomized positions. In view of the perturbation of the velocity, Miao et al. [27] introduced perturbation to the velocity update for the original PSO.

Although the above literatures have considered the importance of velocity and partially modified the velocity formula, they did not consider the case that the history velocities of the particles are ignored. In this paper, we will consider this case for particles to exploit promising areas.

2.2. The PSO Hybrid Algorithm

Many efforts have been made to overcome the weakness of optimization algorithms. It is a common strategy to mix two or more different algorithms to get better performance, e.g., hybridizing differential evolution algorithm with krill herd algorithm [28]. The hybrid purpose is to aggregate the advantages of different algorithms to improve the search ability. A good hybrid algorithm should have a reasonable scheme in allocating exploration and exploitation [8].

Evolutionary algorithm may be the most commonly used tool to mix with the PSO algorithm. A genetic algorithm is a class of evolutionary algorithm, and various genetic operators are usually used. For example, Tawhid et al. [29] combined a genetic arithmetical crossover operator with PSO, and Chen et al. [30] crossed over the personal historical best position of each particle to produce promising exemplars. Differential evolution is another class of evolution algorithm. Parsopoulos et al. [31] used it to control the heuristic parameters for PSO. Zhang et al. [32] applied differential evolution operator to PSO for the bell-shaped mutations on the population diversity. Chen et al. [8] evolved the personal best positions in PSO with differential evolution. In it, two differential mutation operators with different characteristics are adopted: one mutation operator has good global exploration ability, and another mutation operator has good local exploitation ability.

A variety of other meta-heuristic algorithms are applied in the hybrid algorithms. Artificial bee colony is a popular meta-heuristic algorithm, and Vitorino et al. [7] utilized it to produce population diversity when particles fall into local optimum. Cuckoo search is also a swarm intelligent algorithm. Ibrahim et al. [33] incorporated it to PSO, and Huang et al. [34] incorporated the continuous ant colony optimization with PSO and presented four types of hybridization. Additionally, Javidrad et al. [35] presented a PSO algorithm hybridized with simulated annealing algorithm, which is utilized as a local search to improve convergence behavior of PSO.

On the other hand, non-meta-heuristic algorithms are also used to mix with PSO to generate new hybrid algorithms. Luo et al. [36] embedded the gradient descent direction in the velocity update formula of PSO to achieve faster convergence. Wang et al. [37] used two phases: attaining feasible local minima by gradient descent method, and then escaping from the local minimum with the help of PSO. Salajegheh et al. [38] combined first and second order gradient directions with the PSO algorithm to promote the performance and reliability. Derivative-free deterministic approaches are also competitive, and they should receive more attention [39]. Direct search is an important method for solving optimization problems that does not require any information about the gradient of the objective function. They mainly include Pattern Search Algorithm [40], Rosenbrock method [10], and the Mesh Adaptive Search Algorithm [41], etc. Liu et al. [42] proposed the line search to optimize the step-size of the velocity direction of the particle of PSO. Fan et al. [43] applied Nelder–Mead simplex search operator to the top few elite particles and applied PSO to update those particles with worst objective function value. El-Wakeel and Smith [44] also introduced the simplex search to PSO. They first applied PSO to locate the interval that likely contains the global minimum, and then utilized the solution of the PSO as a starting solution for simplex. The PSO algorithm is responsible for avoiding local minima, whereas the simplex algorithm is responsible for avoiding slowness and near-optimum convergence. Tawhid [29] also applied the simplex method as a local search method to accelerate the convergence when no improvements during research in the final stage.

In the above non-meta-heuristic algorithms, Rosenbrock method is a derivative-free deterministic direct search approach. Kang et al. [45] combined the rotational direction of Rosenbrock method to artificial bee colony algorithm, which is utilized in the exploration phase while the direction rotation appeared in the exploitation phase. In this paper, we will make use of the direction search of Rosenbrock method and ignore the direction rotation.

3. Static Exploitation: Searching with Inertia-Free Velocity from the Original PSO

The original PSO is a simulator of social behavior that embodies the movement of a bird's flock. PSO quickly converges due to its inherent parallel ability. Each particle moves towards its best previous position and towards the best particle in the entire swarm. Suppose that the search space is D-dimensional, and a swarm consisting of N particles search in it (i.e., the i-th particle is a D-dimensional vector). In the current work, we consider the PSO global version, in which the best position to be ever attained by all individuals of the swarm is communicated to all the particles at each iteration. The i-th particle updates its position according to the following equations:

$$v_i^{n+1} = \omega \cdot v_i^n + c_1 \cdot rand_1^n(pbest_i^n - x_i^n) + c_2 \cdot rand_2^n(gbest^n - x_i^n), \quad (1)$$

$$x_i^{n+1} = x_i^n + v_i^{n+1}, \quad (2)$$

where $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$ represents the position of the i-th particle; $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ represents the velocity for the i-th particle; $pbest_i$ represents the best previous position of the i-th particle; where $i = 1, 2, \ldots, N$, and N is the size of the swarm, and $n = 1, 2, \ldots,$ is the generation number; $gbest$ represents the best previous position of the population; c_1 and c_2 are the acceleration constants. $rand_1$ and $rand_2$ are two random numbers in the range [0, 1], and ω is the inertia weight.

The velocity update Formula (1) consists of three parts: part 1 is the inertia velocity of the particle, which may balance the global and local search ability; part 2 is the cognitive part, which indicates that the particles think and allows them to have a sufficiently strong global search ability; part 3 acts as a social part, which shows the information sharing and cooperation between the particles. The three parts together determine the space search ability of the particle.

The above formula is a general method to update a particle's position in the original PSO method. In part 1 of Equation (1), the inertia weight ω may control the flying velocity and balance the global and local search [16]. A large value of the weight encourages the global search while a small value enhances the local search [46]. However, it does not seem that there is an accurate parameter (ω) to identify this effect.

If the weight velocity item is cancelled, i.e., let ω = 0, then the particle velocity of the previous generation (history velocity) takes effect no longer. That is to say, the kinetic energy of the particle is greatly reduced after it reaches a point (x_i^S). At this time, the particle is controlled by only the previous best position of itself and the population (i.e., $pbest_i^{(n,t)}$ and $gbest^n$). At this time, the velocity of the particle becomes

$$v_i^{(n,t)} = c_1 \cdot rand_1^n \cdot (pbest_i^n - x_i^n) + c_2 \cdot rand_2^n \cdot (gbest^n - x_i^n),\ t = 1, 2, \ldots, T. \quad (3)$$

Equation (3) suggests that the particles fly without with inertia. Thus, we call the process "static exploitation" (SE). Where $v_i^{(n,t)}$ denotes the velocity gained by the t-th SE of the i-th particle at generation n. Because of the decrease of kinetic energy, the particles may exploit a few positions in a small scope around the point x_i^S. Let T be the maximum exploiting times.

$$x_i^{(n,t)} = x_i^S + v_i^{(n,t)},\ t = 1, 2, \ldots, T. \quad (4)$$

where $x_i^{(n,t)}$ denotes the position gained by the t-th SE of the i-th particle at generation n. Let $x^S = x_i^{n+1}$ as the exploiting center. The best point $x_i^{D_0}$ will be selected in the T exploitations:

$$x_i^{D_0} = \text{argmin} f(x_i^{(n,t)}),\ t = 1, 2, \ldots, T. \tag{5}$$

According to our experiments, excessive exploiting times for trial points do not yield a better effect, generally let $T = 10$.

The attraction of the individual best historical position and the global best position is strengthened when the particles move without inertia, and faster velocity is flattened, which implies that the search on a local area is strengthened.

4. Searching Based on the Rosenbrock Method

4.1. The Rosenbrock Method

The Rosenbrock method [10,47] is a gradient-free minimization direct search method, which is based on orthogonal search directions.

The Rosenbrock method like other direct search methods make a promising descent direction and have surprisingly sound heuristics. It often avoids the pitfalls that plague more sophisticated approaches [48]. The Rosenbrock method can take advantage of a nonzero step and particularly, a promising descent direction along which the next search stage will be conducted, which is apt for searching heuristically at the bottom of a valley.

Rosenbrock's search works through two main search procedures. One is ***direction search* (DS)**, an exploration by discrete steps along an orthogonal direction set of n vectors, and the other is a new search direction set generation and ***rotation procedure*** of the direction set, which is generated using the Gram-Schmidt orthonormalization.

The flow chart of the Rosenbrock method is shown in Figure 1. x_i^S is the initial point of the Rosenbrock search.

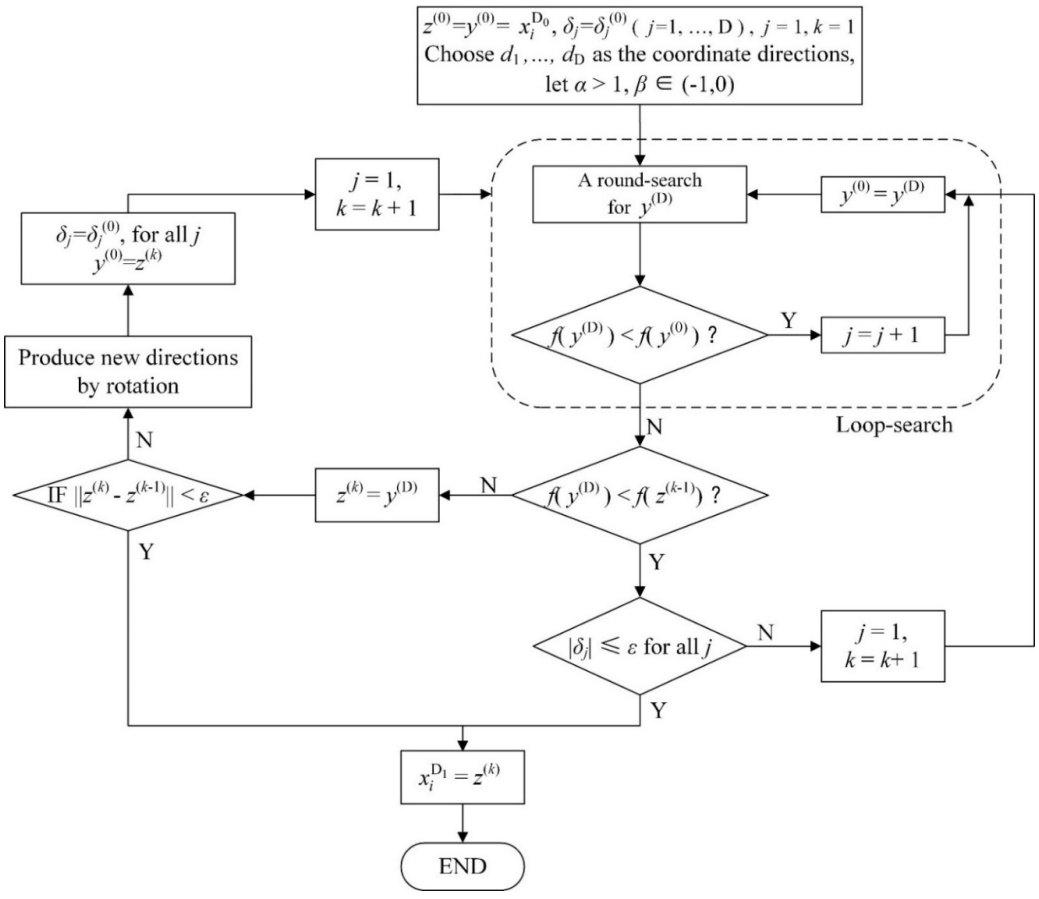

Figure 1. The flow chart of the Rosenbrock method.

The DS procedure involves the round-search and the loop-search. A "round" includes D times searching along D coordinate axes. $y^{(0)}$, $y^{(D)}$ denote the start and the end point of the round, and $y^{(j)}$ denotes the attainted point on the j-th axis, $j \in \{1,2, \ldots, D\}$. This search is always proceeding along the coordinate axes, which are search directions, and cycles until it fails in finding a better point. Thus, a "loop-search" consists one or more consecutive round-searches (Figure 2).

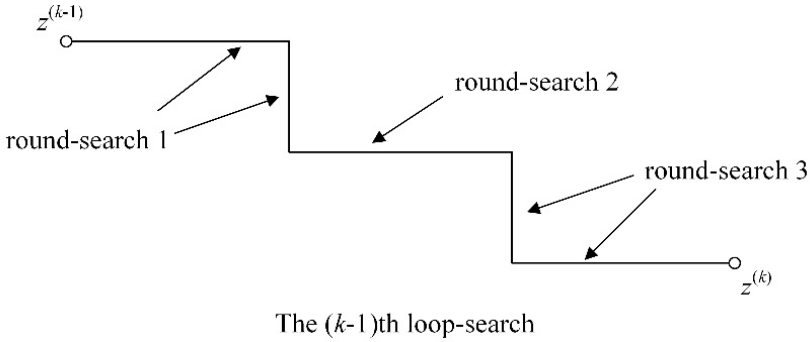

Figure 2. A loop-search is composed of round-searches (with 2 coordinate axes).

After a loop-search, proceeding with loop-search and rotating coordinate axes are two choices. The rotation happens (i.e., a new set of search directions is formed) when at least one success is obtained during a loop search, i.e., at least one point attained by the round-search is better than the starting point of the loop-search. The orthonormal basis is usually updated by the Gram–Schmidt procedure. Let $z^{(k)}(k \in \{1,2,\ldots,\})$ be the end point of the kth loop-search.

4.2. The Procedure of the Round-Search

The directions for the round-search are the coordinate directions of the D-dimensional coordinate system, i.e., the orthonormal basis $d^{(j)}(j = 1,2,\ldots,D)$, which is a group of orthogonal axis directions. These directions are vectors of zeros, except for the unit length 1.0 in the i-th direction.

A round-search starts from $y^{(0)}$ in direction $d^{(1)}$ in direction and then exploits a new point $y^{(j)}$, gained in direction $d^{(j)}$ by the step-side δ_j, until reaching the end point of the round-search $y^{(D)}$ at the last direction $d^{(D)}$. The round-search along the coordinate axis is defined as Algorithm 1.

Algorithm 1: A Round-Search along *D* Coordinate Directions.

```
RoundSearch( )
Set d^(j) (j = 1,..., D) as the coordinate directions and let δ_j^0 (j = 1,..., D) be the initial
step sizes.
{j = 1;
   while (j ≤ D && δ_j > ε) do
   { y = y^(j−1) + δ_j d^(j);
      If f(y) < f(y^(j−1)), let y^(j) = y and δ_j = α · δ_j;//trial step is successful
      Else let y^(j) = y^(j−1) and δ_j = β · δ_j.;//trial step is not successful
   }
    j = j + 1;
}
```

where α is the expansion factor $(\alpha > 1)$, which represents that the step size is increased in this direction, when a successful point is found; β is the constriction factor $(\beta \in (-1,0))$, which represents that the step size is decreased, and the search proceeds on the opposite direction when no point is found at this direction. With the expansion and the constriction of the step size, some valuable points are attained.

4.3. Direction Search (DS, Rosenbrock Procedure without Coordinate Rotation)

The Rosenbrock method is extremely sensitive to the initial point, and easy to get stuck in local minima in many problems by our experiments, and the iterative othonormalization (rotation) procedure of it is time-consuming, increasing the time complexity [49]. However, through the orthogonal search directions (zig-zags in Figure 3), searching can move near a ridge to reach the optimum, which can adapt itself to the local terrain. Simple and heuristic, it performs better on sharp ridges of functions [11].

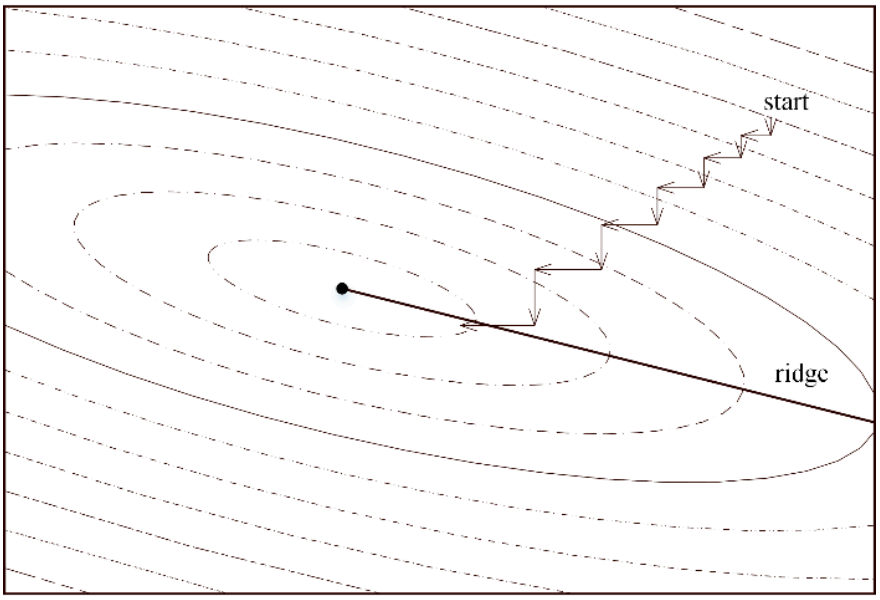

Figure 3. Moving near a ridge for reaching an optimum with direction search (DS) (The concentric ellipses are the contour map of an objective function with two varieties).

Thus, we take the DS procedure from the above Rosenbrock method as a simple component of our hybrid system disregarding the coordinate rotation procedure for intensifying the local optimum search ability of PSO. This procedure is listed in Algorithm 2, where $x_i^{(\mathrm{R})}$ denotes the end point of the procedure.

Algorithm 2: The DS Procedure.

Initialization. $z^{(0)} = y^{(0)} = x_i^S$; set the direction $d^{(j)}$ and the initial step size $\delta_j^{(0)}$ in direction $d^{(j)}$ randomly for each j ($j = 1, 2, \ldots, D$); and let $\alpha > 0$, $\beta \in (0, 1)$.
Repeat
    Call RoundSearch( ) for $y^{(D)}$;
    If ($f(y^{(D)}) < f(y^{(0)})$), then
    { Let $y^{(0)} = y^{(\mathrm{D})}$, call RoundSearch( ) to update $y^{(D)}$;}
    Else
    { If ($f(y^{(D)}) < f(z^{(k-1)})$ //at least a success in loop-search k
            {Break; }
        Else
        { for each δ_j:
                { If $|\delta_j| < \varepsilon$ break;
                Else { $z^{(k)} = y^{(0)} = y^{(D)}, k = k + 1$}.
                }
        }
    }
Step 4. End repeat.
Step 5. Set $x_i^{D_1} = y^{(D)}$.

Algorithm 2 ends when finding a better point than the start point of the loop-search, or each δ_j in different directions is less than a given small value (the termination tolerance) ε; otherwise, the loop-searches composed of the round-searches go on.

The above procedure does not refer to the coordinate rotation for new directions, avoiding the low efficiency of the Rosenbrock method. Moreover, it may lay the foundation of a stable solution due to that it is a deterministic and direct process. Hence, we consider the procedure to be a component of the following algorithm.

5. Proposed Hybrid Particle Swarm Optimization Algorithm

The original PSO tends to present its search advantages in a broad area with its velocity. Then, two additional search stages, SE and DS, follow the original PSO search for exploiting local area. The SE locates a small search area; further, the DS takes a search for this area.

Hence the SE and the DS are incorporated into the original PSO. The SE actually refers to a PSO with inertia-free velocity. So, this algorithm is named hybrid PSO algorithm with inertia-free velocity and the DS (simplified as SDPSO).

The proposed SDPSO algorithm includes three stages: (1) PSO (2) SE (3) DS.

5.1. The Procedure of the SDPSO

The procedure of the proposed SDPSO is outlined in Figure 4.

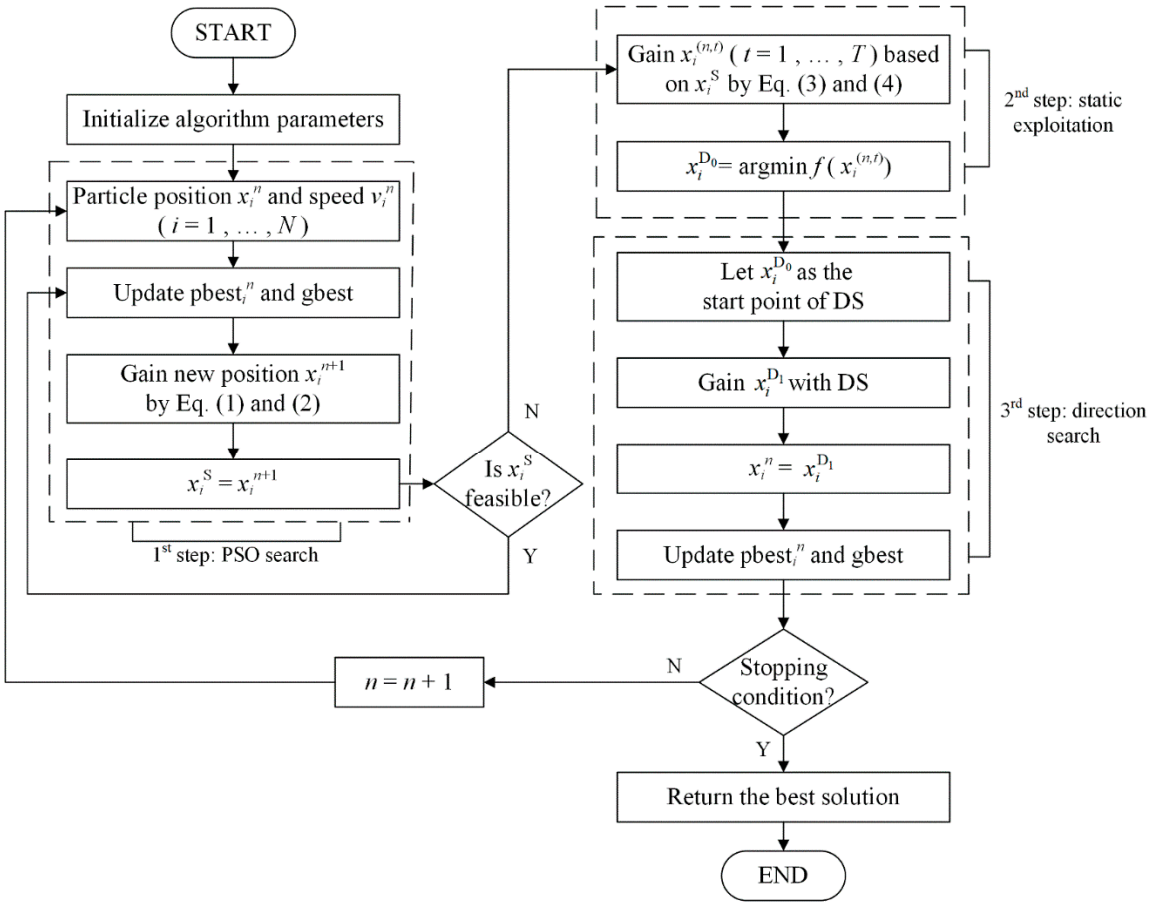

Figure 4. The flowchart of the SDPSO algorithm.

The i-th particle is taken for example to express the process:

Stage 1 (original PSO search): The search of the origin PSO is first executed with Equations (1) and (2) to update particle velocity and position for each particle in each cycle.

Stage 2 (static exploitation, SE): If the i-th particle has flown into an infeasible solution space or convergence stagnation, this particle immediately stops flying and re-

turns to its last feasible position x_i^S, and takes it as the center to exploit several times via Equations (3) and (4). If one or more feasible solutions can be found, select a best one from them and set it to $x_i^{D_0}$ as the start point of the following DS.

Stage 3 (direction search, DS):

Set $x_i^{D_0}$ as an initial point of the DS, searching for a new point through the DS. Then, update $pbest_i^n$ and $gbest^n$.

The SE and the DS are two strengthening search stages. The former exploits the micro-area, while the latter further performs a local search in order to obtain a better solution.

The following simple example illustrates the search procedure of the SDPSO algorithm. Considering the two-dimensional ($D = 2$) problem,

Minimize $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$, $x_1, x_2 \in [0, 3]$.

Figure 5 depicts the search steps at first few generations. Level curves of the function $f(x_1,x_2)$ are shown in the background using shades of gray. The following notations are used in the figure.

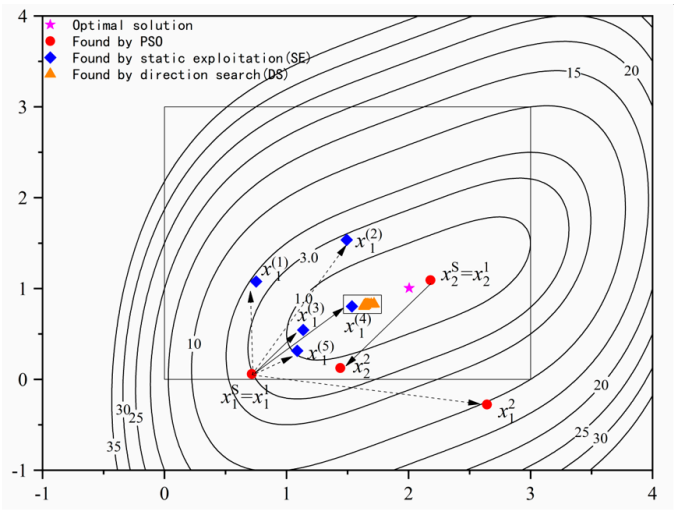

Figure 5. Illustrating the iterations process of the SDPSO. (**a**) represents stage 1 and stage 2. (**b**) represents the stage 3. In (**a**), the big rectangle represents the feasible region; the small rectangle represents the DS and it is enlarged in (**b**). The optimal solution to the problem is (2,1).

x_1^1, x_1^2 the solutions found at the 1th and 2th generation of particle 1 with the origin PSO;

x_2^1, x_2^2 the solutions found at the 1th and 2th generation of particle 2 with the origin PSO;
x_1^S, x_2^S the center points of particle 1 and particle 2 with SE;
$x_1^{(1)}, x_1^{(2)}, x_1^{(3)}, x_1^{(4)}, x_1^{(5)}$ the exploited points of particle 1 with DS.

There are two particles in Figure 5. Produce x_1^1 (0.716, 0.059) and x_2^1 (2.179, 1.092) by initialization.

Stage 1. (see Figure 5a).

The two particles reach x_1^2 (2.642, −0.277) and x_2^2 (1.441, 0.125) by the original PSO.

Stage 2. (see Figure 5a).

For particle 1, because x_1^2 is not a feasible solution, SE performs several random exploitations based on x_1^S (i.e., the last feasible point x_1^1). Five points are exploited, and the best point $x_1^{(4)}$ (1.536, 0.803) is selected as the initial points of DS.

For particle 2, No SE happens because x_2^2 is a feasible point, though x_2^S is assigned a value of x_2^1.

Stage 3. (see Figure 5b).

For particle 1, DS starts from $x_1^{D_0}$ ($x_1^{R_0} = x_1^{(4)}$) till $x_1^{D_1}$, after then, with which the $pbest_1$ or *gbest* is updated.

The above process is on the first generation, and other generations continue as it.

5.2. The Joint Roles of the Three Stages

The three different stages play different roles in during search process. Stage 1 keeps the diversity of the original PSO by a wide range search; Stage 2 (the SE) locates a small local region with a few trials; and stage 3 (the DS) further probes near and along ridges into a micro-region to improve the solution.

The original PSO contributes more to global searches while the other two stages focus more attention on local searches. Furthermore, the DS enforces the solution stability for its non-stochastic performance. Thus, with the three components, the SDPSO algorithm has a comprehensive and joint effect.

6. Experimental Study on Unconstrained and Constrained Optimization Problems

Several experiments on problems from the optimization literature are used to evaluate the approach proposed in Section 5. These experiments include unconstrained benchmark problems (CEC2014) and some constrained engineering design problems, whose solutions obtained by other techniques are available for comparing with and evaluating the proposed approach.

For handling constrains in these experiments on constrained problems, the penalty function method is applied to unfeasible solutions. We add a very high penalty value to the objective function, and the value 1.0×10^{20} was empirically chosen for each experiment on constrained problems. The Problems are listed in Appendix A.

6.1. Experimental Study on Unconstrained Benchmark Problems (CEC2014)

In this section, to check the performance of the SDPSO algorithm on unconstrained benchmark problems, 30 benchmark functions from CEC 2014 are chosen. These functions are shifted or rotated, and most of them are both shifted and rotated. They are complex and can be qualified for evaluating the tested algorithms' characteristics on various problems. The descriptions of these benchmark functions are listed in Table 1.

The parameters of SDPSO are set as: $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 0.3$, $\alpha = 2.0$, $\beta = -0.6$. According to the special session at CEC 2014, we set a maximum FES of 300,000 for the 30-*D* problem. The dimension of the functions is set to 30 and the number of particles is 100.

Table 1. Descriptions of 30 benchmark functions.

	No	Functions	$F_i^* = F_i(x^*)$
Unimodal Functions	1	Rotated High Conditioned Elliptic Function	100
	2	Rotated Bent Cigar Function	200
	3	Rotated Discus Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Ackley's Function	500
	6	Shifted and Rotated Weierstrass Function	600
	7	Shifted and Rotated Griewank's Function	700
	8	Shifted Rastrigin's Function	800
	9	Shifted and Rotated Rastrigin's Function	900
	10	Shifted Schwefel's Function	1000
	11	Shifted and Rotated Schwefel's Function	1100
	12	Shifted and Rotated Katsuura Function	1200
	13	Shifted and Rotated HappyCat Function	1300
	14	Shifted and Rotated HGBat Function	1400
	15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
	16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Function	17	Hybrid Function 1 (N = 3)	1700
	18	Hybrid Function 2 (N = 3)	1800
	19	Hybrid Function 3 (N = 4)	1900
	20	Hybrid Function 4 (N = 4)	2000
	21	Hybrid Function 5 (N = 5)	2100
	22	Hybrid Function 6 (N = 5)	2200
Composition Functions	23	Composition Function 1 (N = 5)	2300
	24	Composition Function 2 (N = 3)	2400
	25	Composition Function 3 (N = 3)	2500
	26	Composition Function 4 (N = 5)	2600
	27	Composition Function 5 (N = 5)	2700
	28	Composition Function 6 (N = 5)	2800
	29	Composition Function 7 (N = 3)	2900
	30	Composition Function 8 (N = 3)	3000
Search Range: $[-100, 100]^D$			

The results of the comparison are shown in the table below, and the test results are expressed as the mean error (Mean) and the standard deviation of the results (STDEV). The mean error value $f(x) - f(x*)$ is used to evaluate the success of the algorithm; the best mean error value is shown in bold. In addition, Wilcoxon rank-sum test [50] are used in this study to compare the mean errors obtained by SDSPO and the other algorithms at the 0.05 level of significance. The statistical significance level of the aggregate results is given in the last three rows of the tables. "−" indicates a case where a compared algorithm shows a poor performance than SDPSO. "+" means that a compared algorithm shows a better performance than SDPSO. "≈" signifies that a compared algorithm and SDPSO are not significantly different.

6.1.1. Comparison 1: SDPSO and Five Standard Algorithms

In this section, 30 benchmark functions with 30 dimensions from CEC 2014 shown in Table 1 are employed for the comparison of SDPSO and five standard algorithms: PSO [51], GA [52], ABC [53], BBO [54] and SA [55]. For each test function, the mean error and standard deviation are calculated over 30 independent runs of each algorithm with FES = 300,000. Comparison results are listed in Table 2.

As shown in Table 2, for unimodal functions (f1–f3), SDPSO shows competitive performance compared with five standard algorithms. For multimodal functions (f4–f16), SDPSO shows better comprehensive performance than PSO, ABC and SA, but overall, BBO performs best. For f4, f11 and f16, the performance of SDPSO closely follows the best performance of GA and BBO. For hybrid functions (f17–f22), SDPSO displays better performance than the other algorithms. For f23, the performance of SDPSO is only worse than that of GA. For f24, the performance of SDPSO is only worse than that of PSO. Finally, based on the statistical results of Wilcoxon rank-sum test, it can be concluded that SDPSO is a highly competitive metaheuristic algorithm variant compared with the five standard algorithms.

Besides, Figure 6 shows some typical convergence curves of SDPSO and PSO for a part of benchmark functions with 30 dimensions from CEC 2014. It can be seen that SDPSO can effectively accelerate convergence though it does not show an obvious advantage in early iterations. This implies that the hybrid system can help the search to jump out of local optimal minima, and SDPSO has the competitive ability of searching the global optimum.

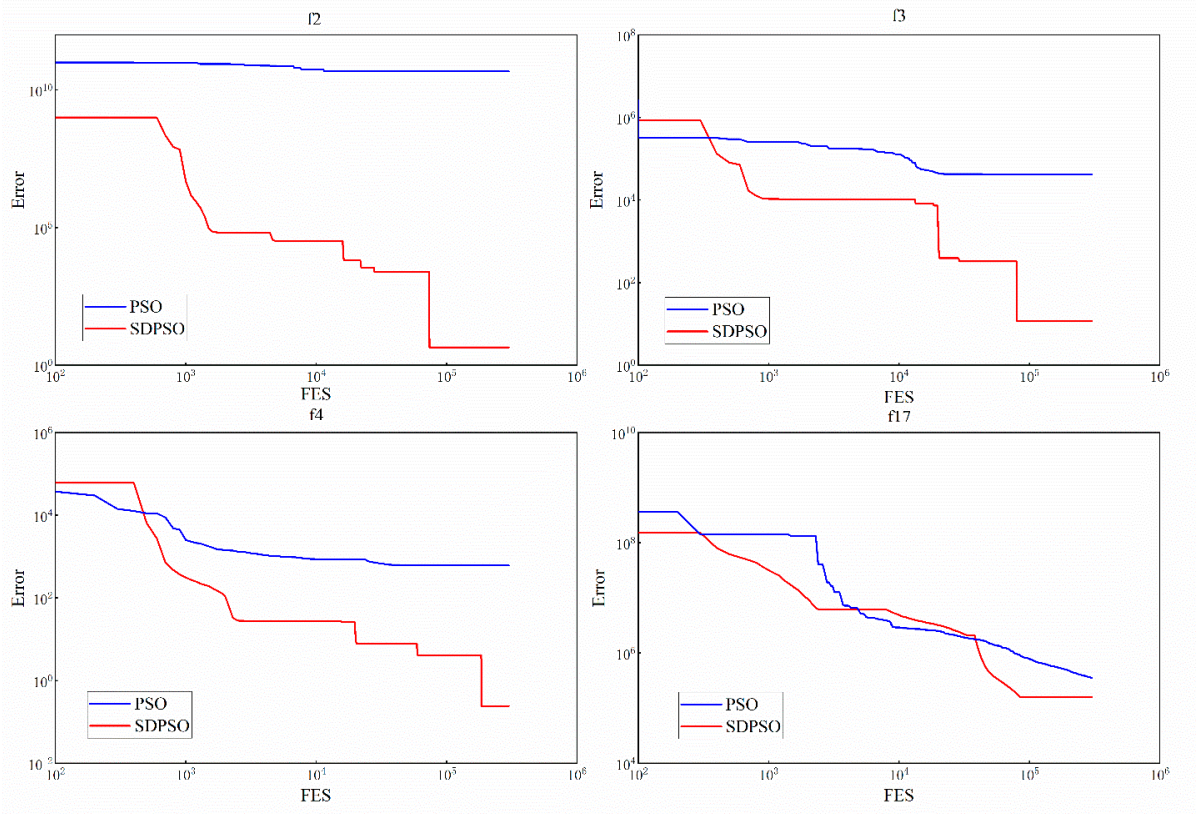

Figure 6. Some typical convergence curves of SDPSO and PSO for a part of bound-constrained benchmark functions with 30 dimensions from CEC 2014.

Table 2. Comparisons of PSO, GA, ABC, BBO, SA and SDPSO over 30 test functions with 30 dimensions.

Fun		PSO	GA	ABC	BBO	SA	SDPSO
F1	Mean	5.41×10^{7} (−)	1.09×10^{6} (−)	6.11×10^{8} (−)	1.95×10^{6} (−)	8.80×10^{8} (−)	2.41×10^{3}
	StdDev	2.24×10^{7}	5.87×10^{5}	1.26×10^{8}	8.26×10^{5}	1.66×10^{8}	2.31×10^{2}
F2	Mean	2.42×10^{8} (−)	7.57×10^{6} (−)	8.57×10^{7} (−)	5.69×10^{4} (−)	7.22×10^{10} (−)	2.07×10^{0}
	StdDev	4.23×10^{8}	2.51×10^{4}	8.25×10^{7}	2.07×10^{4}	3.95×10^{9}	3.89×10^{0}
F3	Mean	5.97×10^{4} (−)	2.14×10^{4} (−)	2.73×10^{5} (−)	1.35×10^{3} (−)	1.25×10^{5} (−)	2.06×10^{1}
	StdDev	1.07×10^{4}	8.27×10^{3}	6.67×10^{4}	1.66×10^{3}	1.01×10^{4}	1.96×10^{1}
F4	Mean	1.66×10^{2} (−)	3.62×10^{0} (+)	2.77×10^{1} (−)	7.60×10^{1} (−)	1.18×10^{4} (−)	3.82×10^{0}
	StdDev	8.29×10^{1}	9.54×10^{-1}	2.37×10^{-1}	4.64×10^{1}	2.62×10^{3}	3.09×10^{0}
F5	Mean	2.08×10^{1} (−)	2.09×10^{1} (−)	2.10×10^{1} (−)	2.00×10^{1} (≈)	2.10×10^{1} (−)	2.00×10^{1}
	StdDev	9.01×10^{1}	6.84×10^{-2}	4.46×10^{-2}	1.58×10^{-2}	5.08×10^{-2}	5.83×10^{-4}
F6	Mean	2.58×10^{1} (+)	2.15×10^{1} (+)	3.93×10^{1} (−)	1.27×10^{1} (+)	3.92×10^{1} (−)	3.40×10^{1}
	StdDev	6.40×10^{1}	1.06×10^{0}	8.87×10^{-1}	4.29×10^{0}	5.93×10^{-1}	2.85×10^{0}
F7	Mean	9.28×10^{1} (−)	1.28×10^{0} (−)	2.65×10^{-1} (−)	2.04×10^{-1} (−)	6.12×10^{2} (−)	1.54×10^{-2}
	StdDev	1.23×10^{1}	1.24×10^{-2}	6.47×10^{-2}	7.46×10^{-2}	4.43×10^{1}	4.85×10^{-3}
F8	Mean	5.09×10^{-2} (+)	1.01×10^{0} (+)	2.01×10^{2} (−)	2.59×10^{1} (+)	2.02×10^{1} (+)	1.59×10^{2}
	StdDev	1.11×10^{0}	9.40×10^{0}	1.08×10^{1}	7.02×10^{0}	1.92×10^{-1}	1.55×10^{1}
F9	Mean	3.18×10^{1} (+)	2.83×10^{1} (+)	2.21×10^{2} (+)	4.95×10^{1} (+)	5.98×10^{1} (+)	2.38×10^{2}
	StdDev	1.28×10^{2}	6.37×10^{0}	1.16×10^{1}	1.43×10^{1}	1.03×10^{1}	3.99×10^{1}
F10	Mean	7.61×10^{2} (+)	8.85×10^{2} (+)	7.15×10^{3} (−)	1.01×10^{3} (+)	1.35×10^{3} (+)	2.45×10^{3}
	StdDev	7.61×10^{2}	4.27×10^{2}	2.79×10^{2}	4.22×10^{2}	1.12×10^{2}	2.80×10^{2}
F11	Mean	6.83×10^{3} (−)	7.53×10^{3} (−)	7.70×10^{3} (−)	3.01×10^{3} (+)	7.23×10^{3} (−)	3.11×10^{3}
	StdDev	1.93×10^{3}	4.25×10^{2}	2.36×10^{2}	6.11×10^{2}	2.66×10^{2}	2.67×10^{2}
F12	Mean	2.86×10^{0} (−)	5.86×10^{-1} (−)	2.48×10^{0} (−)	1.48×10^{-1} (+)	3.02×10^{0} (−)	2.62×10^{-1}
	StdDev	2.77×10^{1}	6.81×10^{-2}	3.39×10^{-1}	6.39×10^{-2}	2.93×10^{-1}	4.31×10^{-2}
F13	Mean	6.40×10^{-1} (−)	2.79×10^{-1} (+)	4.97×10^{-1} (−)	2.15×10^{-1} (+)	6.82×10^{0} (−)	3.21×10^{-1}
	StdDev	4.44×10^{-1}	6.24×10^{-3}	7.15×10^{-2}	5.52×10^{-2}	7.95×10^{-1}	4.88×10^{-2}
F14	Mean	4.21×10^{-1} (−)	2.54×10^{-1} (−)	3.24×10^{-1} (−)	2.31×10^{-1} (−)	2.27×10^{2} (−)	1.86×10^{-1}
	StdDev	1.57×10^{-1}	4.14×10^{-2}	3.70×10^{-2}	4.95×10^{-2}	3.32×10^{1}	2.37×10^{-2}
F15	Mean	4.47×10^{0} (+)	1.06×10^{0} (+)	1.94×10^{1} (−)	5.27×10^{0} (+)	5.24×10^{2} (−)	6.24×10^{0}
	StdDev	1.92×10^{0}	2.59×10^{-2}	1.22×10^{0}	1.07×10^{0}	3.51×10^{2}	9.20×10^{-1}
F16	Mean	1.29×10^{1} (−)	1.30×10^{1} (−)	1.36×10^{1} (−)	1.14×10^{1} (+)	1.31×10^{1} (−)	1.19×10^{1}
	StdDev	4.82×10^{-1}	6.27×10^{-1}	1.33×10^{-1}	6.13×10^{-1}	1.54×10^{-1}	4.78×10^{-1}
F17	Mean	3.26×10^{5} (−)	3.04×10^{5} (−)	8.00×10^{6} (−)	2.60×10^{5} (−)	2.66×10^{6} (−)	4.39×10^{4}
	StdDev	2.57×10^{5}	3.72×10^{4}	2.81×10^{6}	1.66×10^{5}	6.01×10^{5}	2.23×10^{4}
F18	Mean	1.02×10^{6} (−)	5.85×10^{4} (−)	4.71×10^{4} (−)	1.20×10^{3} (−)	1.28×10^{9} (−)	1.89×10^{2}
	StdDev	2.95×10^{6}	6.43×10^{4}	1.10×10^{5}	1.48×10^{3}	4.99×10^{8}	5.19×10^{1}
F19	Mean	2.82×10^{2} (−)	1.62×10^{1} (−)	1.91×10^{1} (−)	1.15×10^{1} (−)	3.03×10^{2} (−)	1.06×10^{1}
	StdDev	1.98×10^{1}	3.52×10^{1}	5.33×10^{-1}	1.04×10^{1}	5.58×10^{1}	1.09×10^{0}
F20	Mean	1.08×10^{4} (−)	2.36×10^{3} (+)	1.16×10^{5} (−)	2.59×10^{3} (+)	6.96×10^{4} (−)	5.08×10^{3}
	StdDev	2.18×10^{3}	3.14×10^{3}	5.49×10^{4}	2.81×10^{3}	2.74×10^{4}	2.41×10^{3}
F21	Mean	7.54×10^{5} (−)	1.34×10^{5} (−)	2.49×10^{6} (−)	1.69×10^{5} (−)	6.23×10^{6} (−)	2.43×10^{4}
	StdDev	4.21×10^{5}	5.59×10^{4}	7.99×10^{5}	1.10×10^{5}	2.82×10^{6}	1.85×10^{4}
F22	Mean	8.24×10^{2} (−)	1.38×10^{3} (−)	7.63×10^{2} (−)	4.02×10^{2} (−)	1.36×10^{3} (−)	2.41×10^{2}
	StdDev	5.24×10^{2}	1.28×10^{2}	1.10×10^{2}	1.78×10^{2}	2.63×10^{2}	8.61×10^{1}
F23	Mean	3.42×10^{2} (−)	3.01×10^{2} (−)	3.37×10^{2} (−)	3.15×10^{2} (−)	7.36×10^{2} (−)	3.14×10^{2}
	StdDev	6.24×10^{0}	3.87×10^{-2}	1.82×10^{0}	1.69×10^{-3}	8.22×10^{1}	3.01×10^{-2}
F24	Mean	2.05×10^{2} (+)	2.51×10^{2} (−)	2.33×10^{2} (−)	2.29×10^{2} (−)	4.08×10^{2} (−)	2.27×10^{2}
	StdDev	2.41×10^{-1}	1.40×10^{1}	6.53×10^{0}	5.89×10^{0}	1.78×10^{1}	7.57×10^{-1}
F25	Mean	2.20×10^{2} (−)	2.18×10^{2} (−)	2.52×10^{2} (−)	2.13×10^{2} (−)	2.70×10^{2} (−)	2.01×10^{2}
	StdDev	4.51×10^{0}	8.47×10^{0}	1.14×10^{1}	4.17×10^{0}	1.04×10^{1}	7.57×10^{-2}
F26	Mean	1.00×10^{2} (≈)	1.56×10^{2} (−)	1.01×10^{2} (−)	1.10×10^{2} (−)	1.01×10^{2} (−)	1.00×10^{2}
	StdDev	2.42×10^{-1}	4.44×10^{1}	5.64×10^{-2}	3.05×10^{1}	3.20×10^{-1}	2.68×10^{-2}
F27	Mean	2.51×10^{3} (−)	7.89×10^{2} (−)	1.30×10^{3} (−)	4.70×10^{2} (−)	9.23×10^{2} (−)	4.03×10^{2}
	StdDev	4.82×10^{2}	2.06×10^{2}	3.82×10^{1}	8.76×10^{1}	1.38×10^{2}	7.61×10^{-1}
F28	Mean	1.81×10^{3} (−)	3.46×10^{3} (−)	4.94×10^{2} (−)	1.31×10^{3} (−)	5.08×10^{3} (−)	4.07×10^{2}
	StdDev	4.91×10^{2}	8.02×10^{3}	2.12×10^{1}	3.92×10^{2}	3.87×10^{2}	8.52×10^{0}
F29	Mean	8.77×10^{7} (−)	1.39×10^{4} (−)	3.55×10^{2} (−)	1.32×10^{3} (−)	1.87×10^{8} (−)	2.07×10^{2}
	StdDev	3.24×10^{7}	1.97×10^{5}	3.46×10^{1}	3.00×10^{2}	1.73×10^{4}	7.21×10^{-1}
F30	Mean	4.11×10^{5} (−)	3.51×10^{3} (−)	1.72×10^{3} (−)	2.68×10^{3} (−)	1.01×10^{6} (−)	4.11×10^{2}
	StdDev	1.87×10^{4}	2.08×10^{3}	1.64×10^{2}	6.47×10^{2}	5.00×10^{5}	7.57×10^{1}
+		6	8	1	10	3	
−		23	22	29	19	27	
≈		1	0	0	1	0	

6.1.2. Comparison 2: SDPSO and PSO Variants

In this section, the comparison of SDPSO and three PSO variants (CLPSO [13], APSO [56] and OLPSO [57]) is conducted. Their mean errors and standard deviations are obtained under 30 independent runs, which are taken from [58] and presented in Table 3. To make a fair comparison, SDPSO is run 30 times for each test function. For all algorithms, the FES = 300,000.

Table 3. Comparisons of CLPSO, APSO, OLPSO and SDPSO over 30 test functions with 30 dimensions.

Fun		CLPSO	APSO	OLPSO	SDPSO
F1	Mean	9.41×10^{6} (−)	1.38×10^{5} (−)	6.12×10^{6} (−)	2.41×10^{3}
	StdDev	3.02×10^{6}	9.59×10^{4}	3.58×10^{6}	2.31×10^{2}
F2	Mean	2.76×10^{2} (−)	4.34×10^{-3} (+)	1.28×10^{3} (−)	2.07×10^{0}
	StdDev	6.84×10^{2}	1.02×10^{-2}	1.48×10^{3}	3.89×10^{0}
F3	Mean	3.24×10^{2} (−)	2.61×10^{2} (−)	3.23×10^{2} (−)	2.06×10^{1}
	StdDev	2.79×10^{2}	4.10×10^{2}	5.69×10^{2}	1.96×10^{1}
F4	Mean	8.07×10^{1} (−)	6.85×10^{0} (−)	8.64×10^{1} (−)	3.82×10^{0}
	StdDev	1.58×10^{1}	2.03×10^{1}	2.22×10^{1}	3.09×10^{0}
F5	Mean	2.05×10^{1} (−)	2.00×10^{1} (≈)	2.03×10^{1} (−)	2.00×10^{1}
	StdDev	4.95×10^{-2}	1.88×10^{-4}	1.28×10^{-1}	5.83×10^{-4}
F6	Mean	1.43×10^{1} (+)	1.58×10^{1} (+)	5.09×10^{0} (+)	3.40×10^{1}
	StdDev	1.38×10^{0}	3.53×10^{0}	1.48×10^{0}	2.85×10^{0}
F7	Mean	6.68×10^{-5} (+)	1.73×10^{-2} (−)	1.02×10^{-13} (+)	1.54×10^{-2}
	StdDev	5.86×10^{-5}	2.08×10^{-2}	3.47×10^{-14}	4.85×10^{-3}
F8	Mean	3.95×10^{-11} (+)	8.86×10^{-12} (+)	0.00×10^{0} (+)	1.59×10^{2}
	StdDev	5.48×10^{-11}	4.67×10^{-11}	0.00×10^{0}	1.55×10^{1}
F9	Mean	6.11×10^{1} (+)	9.13×10^{1} (+)	4.06×10^{1} (+)	2.38×10^{2}
	StdDev	7.92×10^{0}	2.46×10^{1}	7.02×10^{0}	3.99×10^{1}
F10	Mean	3.13×10^{0} (+)	7.92×10^{-1} (+)	8.72×10^{-2} (+)	2.45×10^{3}
	StdDev	1.52×10^{0}	8.25×10^{-1}	2.04×10^{-1}	2.80×10^{2}
F11	Mean	2.87×10^{3} (+)	2.74×10^{3} (+)	2.28×10^{3} (+)	3.11×10^{3}
	StdDev	2.73×10^{2}	5.37×10^{2}	4.66×10^{2}	2.67×10^{2}
F12	Mean	5.38×10^{-1} (−)	1.95×10^{-1} (+)	2.28×10^{-1} (+)	2.62×10^{-1}
	StdDev	7.21×10^{-2}	7.17×10^{-2}	6.38×10^{-2}	4.31×10^{-2}
F13	Mean	3.32×10^{-1} (−)	4.33×10^{-1} (−)	2.59×10^{-1} (+)	3.21×10^{-1}
	StdDev	3.46×10^{-2}	9.22×10^{-2}	3.20×10^{-2}	4.88×10^{-2}
F14	Mean	2.78×10^{-1} (−)	3.23×10^{-1} (−)	2.41×10^{-1} (−)	1.86×10^{-1}
	StdDev	2.98×10^{-2}	1.10×10^{-1}	2.66×10^{-2}	2.37×10^{-2}
F15	Mean	8.62×10^{0} (−)	2.96×10^{1} (−)	6.67×10^{0} (−)	6.24×10^{0}
	StdDev	1.09×10^{0}	4.03×10^{0}	1.62×10^{0}	9.20×10^{-1}
F16	Mean	1.06×10^{1} (+)	1.05×10^{1} (+)	1.17×10^{1} (+)	1.19×10^{1}
	StdDev	3.80×10^{-1}	8.21×10^{-1}	5.48×10^{-1}	4.78×10^{-1}
F17	Mean	8.59×10^{5} (−)	3.19×10^{4} (+)	7.98×10^{5} (−)	4.39×10^{4}
	StdDev	3.58×10^{5}	2.14×10^{4}	4.13×10^{5}	2.23×10^{4}
F18	Mean	1.69×10^{2} (+)	3.73×10^{3} (−)	3.58×10^{2} (−)	1.89×10^{2}
	StdDev	5.85×10^{1}	5.23×10^{3}	5.12×10^{2}	5.19×10^{1}
F19	Mean	8.35×10^{0} (+)	1.41×10^{1} (−)	6.13×10^{0} (+)	1.06×10^{1}
	StdDev	8.04×10^{-1}	1.84×10^{1}	8.20×10^{-1}	1.09×10^{0}
F20	Mean	3.26×10^{3} (+)	6.38×10^{3} (−)	5.58×10^{3} (−)	5.08×10^{3}
	StdDev	1.70×10^{3}	4.86×10^{3}	4.01×10^{3}	2.41×10^{3}
F21	Mean	8.08×10^{4} (−)	2.16×10^{4} (+)	1.07×10^{5} (−)	2.43×10^{4}
	StdDev	3.99×10^{4}	1.31×10^{4}	8.33×10^{4}	1.85×10^{4}
F22	Mean	1.75×10^{2} (+)	6.50×10^{2} (−)	2.20×10^{2} (+)	2.41×10^{2}
	StdDev	7.77×10^{1}	2.42×10^{2}	1.07×10^{2}	8.61×10^{1}
F23	Mean	3.15×10^{2} (−)	3.15×10^{2} (−)	3.15×10^{2} (−)	3.14×10^{2}
	StdDev	4.86×10^{-5}	1.15×10^{-12}	1.23×10^{-10}	3.01×10^{-2}
F24	Mean	2.25×10^{2} (+)	2.29×10^{2} (−)	2.24×10^{2} (+)	2.27×10^{2}
	StdDev	1.29×10^{0}	4.73×10^{0}	5.47×10^{-1}	7.57×10^{-1}
F25	Mean	2.08×10^{2} (−)	2.16×10^{2} (−)	2.09×10^{2} (−)	2.01×10^{2}
	StdDev	1.19×10^{0}	5.81×10^{0}	1.75×10^{0}	7.57×10^{-2}

Table 3. *Cont.*

Fun		CLPSO	APSO	OLPSO	SDPSO
F26	Mean	1.00×10^2 (≈)	1.58×10^2 (−)	1.00×10^2 (≈)	1.00×10^2
	StdDev	7.35×10^{-2}	6.05×10^1	4.44×10^{-2}	2.68×10^{-2}
F27	Mean	4.17×10^2 (−)	6.84×10^2 (−)	3.26×10^2 (+)	4.03×10^2
	StdDev	5.24×10^0	2.11×10^2	3.80×10^1	7.61×10^{-1}
F28	Mean	8.98×10^2 (−)	2.53×10^3 (−)	8.73×10^2 (−)	4.07×10^2
	StdDev	5.32×10^1	8.16×10^2	2.97×10^1	8.52×10^0
F29	Mean	1.29×10^3 (−)	1.24×10^3 (−)	1.36×10^3 (−)	2.07×10^2
	StdDev	1.69×10^2	5.02×10^2	2.82×10^2	7.21×10^{-1}
F30	Mean	3.63×10^3 (−)	2.50×10^3 (−)	2.39×10^3 (−)	4.11×10^2
	StdDev	1.00×10^3	6.63×10^2	5.99×10^2	7.57×10^1
+		12	10	13	
−		17	19	16	
≈		1	1	1	

As shown in Table 3, for unimodal functions (f1–f3) and composition functions (f23–f30), SDPSO is fully superior to the other three algorithms except for very few results from f2, f24 and f27, though the composition function can have different properties for different variables subcomponents.

For simple multimodal functions f13–f15, SDPSO outperforms other three algorithms. For Hybrid Function f17–f22, although CLPSO have advantages over SDPSO, SDPSO has better performance than APSO and OLPSO.

The Shifted and Rotated Rosenbrock's Function (f4) has a very narrow valley from local optimum to global optimum [59], but SDPSO solves it better than the others, which suggests that the DS provides help to jump out of local optimal regions. However, for f6–f12 of simple multiple functions, SDPSO is worst among the four PSOs, the possible reason is that their local optima's number are huge and second better local optimum is far from the global optimum [59].

For the functions with 30 dimensions, SDPSO outperforms the others on a majority of the functions. Further, in order to check the performance of SDPSO in higher-dimensional functions, the dimension of the functions is set to 100 and the results are compared with other state-of-the-art PSO variants, Switch-PSO [60], S-PSO [61], AIW-PSO [62] and DLI-PSO [63].

The parameters of SDPSO are set as: $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 0.3$, $\alpha = 2.0$, $\beta = -0.6$, and the number of particles is 100. For the sake of fairness, the same maximum FES (200,000) with the Switch-PSO, S-PSO, AIW-PSO and DLI-PSO is set for each function of the CEC 2014. The average performances of the five algorithms are tabulated in Table 4.

As shown in Table 4, from the 30 functions tested, the SDPSO is fully superior to the DLI-PSO algorithm in all functions and the other three algorithms in 26 functions. Thus, the performance of SDPSO in higher dimensions is outstanding in comparison to the other state-of-the-art PSO variants and has the potential to solve higher dimensional problems. Generally, the SDPSO is observed to be a good algorithm for solving the unconstrained benchmark problems.

Table 4. Comparisons of Switch-PSO, S-PSO, AIW-PSO, DLI-PSO and SDPSO over 30 test functions with 100 dimensions.

Fun	Switch-PSO	S-PSO	AIW-PSO	DLI-PSO	SDPSO
F1	1.43×10^{8} (−)	2.43×10^{8} (−)	2.05×10^{8} (−)	1.43×10^{10} (−)	8.92×10^{3}
F2	2.49×10^{7} (−)	4.27×10^{7} (−)	5.31×10^{5} (−)	5.59×10^{11} (−)	7.08×10^{3}
F3	7.12×10^{4} (−)	9.86×10^{4} (−)	4.08×10^{4} (−)	8.40×10^{5} (−)	1.85×10^{4}
F4	1.03×10^{3} (−)	1.14×10^{3} (−)	1.15×10^{3} (−)	2.33×10^{5} (−)	7.30×10^{1}
F5	5.21×10^{2} (−)	5.21×10^{2} (−)	5.21×10^{2} (−)	5.21×10^{2} (−)	2.00×10^{1}
F6	6.72×10^{2} (−)	6.84×10^{2} (−)	6.84×10^{2} (−)	7.66×10^{2} (−)	1.57×10^{2}
F7	7.01×10^{2} (−)	7.01×10^{2} (−)	7.00×10^{2} (−)	5.80×10^{3} (−)	2.44×10^{-2}
F8	1.01×10^{3} (+)	1.01×10^{3} (+)	9.93×10^{2} (+)	2.73×10^{3} (−)	1.05×10^{3}
F9	1.24×10^{3} (+)	1.33×10^{3} (+)	1.35×10^{3} (+)	3.20×10^{3} (−)	1.47×10^{3}
F10	6.95×10^{3} (+)	7.72×10^{3} (+)	6.39×10^{3} (+)	3.34×10^{4} (−)	1.35×10^{4}
F11	1.46×10^{4} (−)	2.59×10^{4} (−)	1.55×10^{4} (−)	3.35×10^{4} (−)	1.45×10^{4}
F12	1.20×10^{3} (−)	1.20×10^{3} (−)	1.20×10^{3} (−)	1.20×10^{3} (−)	5.00×10^{-1}
F13	1.30×10^{3} (−)	1.30×10^{3} (−)	1.30×10^{3} (−)	1.31×10^{3} (−)	5.06×10^{-1}
F14	1.40×10^{3} (−)	1.40×10^{3} (−)	1.40×10^{3} (−)	2.85×10^{3} (−)	3.39×10^{-1}
F15	1.57×10^{3} (−)	1.60×10^{3} (−)	1.59×10^{3} (−)	3.76×10^{8} (−)	6.15×10^{1}
F16	1.64×10^{3} (−)	1.65×10^{3} (−)	1.65×10^{3} (−)	1.65×10^{3} (−)	4.51×10^{1}
F17	1.02×10^{7} (−)	2.60×10^{7} (−)	3.09×10^{7} (−)	1.75×10^{9} (−)	1.04×10^{4}
F18	3.39×10^{3} (−)	2.05×10^{5} (−)	7.17×10^{5} (−)	5.87×10^{10} (−)	3.08×10^{3}
F19	2.07×10^{3} (−)	2.09×10^{3} (−)	2.08×10^{3} (−)	1.53×10^{4} (−)	5.68×10^{1}
F20	5.55×10^{4} (+)	6.70×10^{4} (+)	5.23×10^{4} (+)	2.18×10^{7} (−)	9.39×10^{4}
F21	5.45×10^{6} (−)	1.23×10^{7} (−)	1.04×10^{7} (−)	9.19×10^{8} (−)	1.11×10^{4}
F22	4.38×10^{3} (−)	4.78×10^{3} (−)	4.65×10^{3} (−)	7.40×10^{5} (−)	2.14×10^{3}
F23	2.66×10^{3} (−)	2.66×10^{3} (−)	2.66×10^{3} (−)	9.78×10^{3} (−)	3.45×10^{2}
F24	2.80×10^{3} (−)	2.80×10^{3} (−)	2.79×10^{3} (−)	4.13×10^{3} (−)	4.20×10^{2}
F25	2.77×10^{3} (−)	2.80×10^{3} (−)	2.79×10^{3} (−)	3.99×10^{3} (−)	2.03×10^{2}
F26	2.80×10^{3} (−)	2.81×10^{3} (−)	2.81×10^{3} (−)	3.87×10^{3} (−)	1.01×10^{2}
F27	4.77×10^{3} (−)	5.12×10^{3} (−)	5.22×10^{3} (−)	7.85×10^{3} (−)	1.29×10^{3}
F28	7.10×10^{3} (−)	9.87×10^{3} (−)	1.05×10^{4} (−)	2.91×10^{4} (−)	5.48×10^{2}
F29	7.45×10^{3} (−)	1.40×10^{4} (−)	6.93×10^{3} (−)	3.04×10^{9} (−)	2.50×10^{2}
F30	8.06×10^{4} (−)	1.61×10^{5} (−)	2.04×10^{5} (−)	1.80×10^{8} (−)	2.51×10^{3}
+	4	4	4	0	
−	26	26	26	30	
≈	0	0	0	0	

6.2. Experimental Study on Constrained Engineering Design Problems

In order to verify the performance of the SDPSO on constrained optimization problems, the experimental results of a few design optimization problems are compared with some state-of-the-art algorithms.

The parameters of SDPSO are set as: $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 0.5$, $\alpha = 3.0$, $\beta = -0.5$. The number of particles is 50. Each experiment is independently run 100 times for all the compared algorithms. The number of function evaluations (FES) is different with the problems, since the comparative results that come from different literatures have different FES.

6.2.1. Pressure Vessel Design Optimization Problem (Problem 1)

Many optimization algorithms and variants have been applied to solve this problem, such as diversity-enhanced constrained PSO (DEC-PSO) [64], coevolutionary particle swarm optimization (CPSO) [65], hybrid PSO (HPSO) [66], multi-population GA (BIANCA) [67], firefly algorithm (FFA) [68], PSO-DE [69], passing vehicle search (PVS) [70], artificial bee colony algorithm (ABC) [53], constraint violation with interval arithmetic PSO (CVI-PSO) [71], bat algorithm (BA) [72], teaching-learning-based optimization (TLBO) [73], Hybrid Nelder-Mead simplex search and particle swarm optimization (NM-PSO) [74].

Because the results of these algorithms come from different literatures, and the corresponding FES for the results is different. For the SDPSO, the results at 20,000 FES and

42,100 FES are recorded for comparison, respectively (Tables 5 and 6). For a comparable situation, the comparative results with the above literatures are summarized in the two tables by the different FES.

Table 5. Statistical results of different methods for problem 1 (20,000 FES for SDPSO).

Algorithm	Best	Mean	Worst	FES
ABC [53]	6059.714	6245.308	NA	30,000
CVI-PSO [71]	6059.714	6292.123	6820.41	25,000
BA [72]	6059.714	6179.13	6318.95	20,000
TLBO [73]	6059.714	6059.714	NA	20,000
PVS [70]	6059.714	6065.877	6090.526	20,000
SDPSO	**5885.902**	**5906.450**	**6069.794**	20,000

(NA means not available).

Table 6. Statistical results of different methods for problem 1 (42,000 FES for SDPSO).

Algorithm	Best	Mean	Worst	FES
DEC-PSO [64]	6059.714	6060.33	6090.526	300,000
CPSO [65]	6061.0777	6147.1332	6363.8041	240,000
HPSO [66]	6059.7143	6099.9323	6288.6770	81,000
BIANCA [67]	6059.938	6182.002	6447.325	80,000
FFA [68]	6059.714	6064.33	6090.52	50,000
PSO-DE [69]	6059.714	6059.714	6059.714	42,100
PVS [70]	6059.714	6063.643	6090.526	42,100
SDPSO	**5885.378**	**5885.881**	**5886.373**	42,100

From Table 5, all the statistical results (the best, the mean, and the worst result of the objective function values) of the SDPSO algorithm are much better than those of the other algorithms. The optimal solution for the 5885.902 is (0.778464074, 0.384810342, 40.33477797, 199.7890799).

At the case that FES = 42,100 (Table 6), the SDPSO also keeps the best performance in terms of the best, mean and worst values. The SDPSO not only finds the new global solution on the problem, but also the best, mean and worst values are much smaller than those of the other algorithms. Hence, it can be concluded that the SDPSO is more efficient than the other algorithms for the pressure vessel design problem, and the optimal solution for the 5885.378 is (0.778177268, 0.384652711, 40.31982465, 199.9971357).

6.2.2. Speed Reducer Design Optimization Problem (Problem 2)

This problem has been solved by society and civilization method (SCM) [75], accelerating adaptive trade-off model (AATM) [76], differential evolution with level comparison (DELC) [77], multi-view differential evolution algorithm (MVDE) [78], passing vehicle search (PVS) [70]. This problem is solved by the SDPSO with 30,000 FES, and the results are shown in Table 7.

Table 7. Statistical results of different methods for problem 2 (30,000 FES for SDPSO).

Algorithm	Best	Mean	Worst	FES
SCM [75]	2994.744241	3001.758264	3009.964736	54,456
AATM [76]	2994.516778	2994.585417	2994.659797	40,000
DELC [77]	**2994.471066**	**2994.471066**	**2994.471066**	30,000
MVDE [78]	**2994.471066**	**2994.471066**	2994.471069	30,000
PVS [70]	**2994.471066**	2994.472059	2994.477593	30,000
SDPSO	2994.471067	2994.471081	2994.471166	30,000

As shown in Table 7, it is obvious that the best value (2994.471067) found by the SDPSO is close to the best value (2994.471066) found by the DELC, MVDE and PVS. For

the mean, the value 2994.471081 found by the SDPSO ranks the second, and just slightly worse than the best mean (2994.471066). For the worst value, the SDPSO has also ranked among the best three algorithms.

6.2.3. Spring Design Optimization Problem (Problem 3)

For this problem, the SDPSO compares with 11 different algorithms. Because the comparative results come from different literatures, their FES are different. Thus, their results are listed into two tables according to different FES, and the results of the SDPSO are given at 20,000 and 42,100 FES (see Tables 8 and 9).

Table 8. Statistical results of different methods for problem 3 (20,000 FES for SDPSO).

Algorithm	Best	Mean	Worst	FES
ABC [53]	0.012665	0.012709	NA	30,000
CVI-PSO [71]	0.012666	0.012731	0.012843	25,000
BA [72]	**0.012665**	0.013501	0.016895	20,000
PVS [70]	**0.012665**	**0.012666**	**0.012667**	20,000
TLBO [73]	**0.012665**	**0.012666**	NA	20,000
SDPSO	**0.012665**	0.012703	0.013187	20,000

(NA means not available).

Table 9. Statistical results of different methods for problem 3 (42,000 FES for SDPSO).

Algorithm	Best	Mean	Worst	FES
CPSO [65]	0.012674	0.012730	0.012924	240,000
HPSO [66]	**0.012665**	0.012707	0.012719	81,000
NM–PSO [74]	0.012630	0.012631	0.012633	80,000
BIANCA [67]	0.012671	0.012681	0.012913	80,000
FFA [68]	0.012665	0.012677	0.013000	50,000
PSO-DE [69]	**0.012665**	**0.012665**	**0.012665**	42,100
PVS [70]	**0.012665**	**0.012665**	**0.012665**	42,100
SDPSO	**0.012665**	**0.012665**	0.012668	42,100

For the best value, the SDPSO can find it before 20,000 FES as the other algorithms (Table 8). For the mean value, the SDPSO falls into the top category at 42,100 FES (Table 9), though it is not the best at 20,000 FES. For the worst value, in Table 7, 0.012668 is slightly less than the first place (0.012665) at 42,100 FES.

6.2.4. Welded Beam Design Problem (Problem 4)

For the welded beam design problem, the results of the SDPSO are compared with those of the following four algorithms: society and the civilization model (SCM) [75], hybrid real-parameter genetic algorithm (ARSAGA) [79], differential evolution with dynamic stochastic selection (DSS-MDE) [80] and a multiagent evolutionary optimization algorithm (RAER) [81]. The comparison results are shown in Table 10.

Table 10. Statistical results of different methods for problem 4.

Algorithm	Best	Mean	Worst	FES
SCM [75]	2.3854347	3.2551371	6.3996785	33,095
ARSAGA [79]	NA	2.25	NA	26,466
DSS-MDE [80]	**2.38095658**	**2.38095658**	**2.38095658**	24,000
RAER [81]	2.38117	2.38117	2.3812	18,467
SDPSO	2.381017466	2.412894742	2.888707	33,000

(NA means not available).

Although the average and worst value of the SDPSO is not as good as the other algorithms in the table above for the welded beam design problem, the best value (2.381017466)

it finds ranks second and is very close to the best result (2.38095658) attainted by DSS-MED in the table.

6.2.5. Three-Bar Truss Design Problem (Problem 5)

For this problem, the results of the SDPSO are compared with those of the 7 algorithms.

From Table 11, the best value of the SDPSO almost reaches the best, and the mean and the worst of it rank second and almost also attain the best result of the other algorithms, even though the performance of it is not outstanding in these results of the comparative algorithms.

Table 11. Statistical results of different methods for problem 5.

Algorithm	Best	Mean	Worst	FES
SCM [75]	263.8958465	263.9033567	263.9697564	17,610
PSO-DE [69]	**263.8958434**	**263.8958434**	**263.8958434**	17,600
AATM [76]	263.8958435	263.8966	263.90041	17,000
DSS-MDE [80]	**263.8958434**	263.8958436	263.8958498	15,000
MVDE [78]	**263.8958434**	**263.8958434**	263.8958548	7,000
SDPSO	263.8958435	263.8966668	263.9023268	15,000

Based on the above comparisons, it can be concluded that the SDPSO sustains competitiveness on solving constrained engineering design problems.

7. Parameters Study of SDPSO

Shi, Y. and Eberhart, R.C. [82] have found that the original PSO algorithm can get a better performance with the inertia weight ω in the range [0.9, 1.2], and the acceleration factors c_1 and c_2 fixed at 2.0 in the early experiments [51,83]. Later studies analyzed the relationship between the parameters, for its convergence, to set $c_1 = c_2 = 1.49445$, $\omega = 0.729$ [84]. However, SDPSO has taken a different strategy from the original PSO, which may produce a different influence from ω. Parameters should be well selected by experiments for good performance. In this paper, seven traditional unconstrained benchmark test functions [23] in Table 12 were adopted to test for finding a suitable scope of ω, c_1 and c_2 for a stable performance. $\alpha = 3.0$ and $\beta = -0.5$ are from [10]. They are moderate and used in SDPSO.

Table 12. Unconstrained benchmark test functions with 30 dimensions.

Fun	Functions	Domain	Best
f1	$f_1(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	$(-10, 10)$	-78.3323
f2	$f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$(-10, 10)$	0
f3	$f_3(x) = \sum_{i=1}^{n} x_i^2$	$(-10, 10)$	0
f4	$f_4(x) = \sum_{i=1}^{n-1}(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$	$(-2.048, 2.048)$	0
f5	$f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$	$(-1, 1)$	0
f6	$f_6(x) = \sum_{i=1}^{n} ix_i^2$	$(-10, 10)$	0
f7	$f_7(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	$(-3, 3)$	0

7.1. Impacts of Inertia Weight on SDPSO

For testing the impact of inertia weight w, it is chosen from 0.1 to 2.0 with 0.05 steps. Let $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 3.0$, $\alpha = 3.0$, $\beta = -0.5$, and the swarm size of 100. Each run reached the known solution with an error of 0.00001.

The average function evaluation value with different ω is drawn in Figure 7. In the figure, the best ω is generally located between 0.1 and 0.6. For a higher value, the performance of the algorithm has degraded from the Figure 7. It is different from the original PSO, where the inertia weight ω is usually larger [84]. Shi and Eberhart [82] have stated that the inertia weight is responsible for balancing between the local and global search abilities, and the small inertia weight facilitates local search while the large inertia weight facilitates global search. Therefore, with the lower values of ω in SDPSO, local search is intensified, which may be due to the introduction of the SE and the DS.

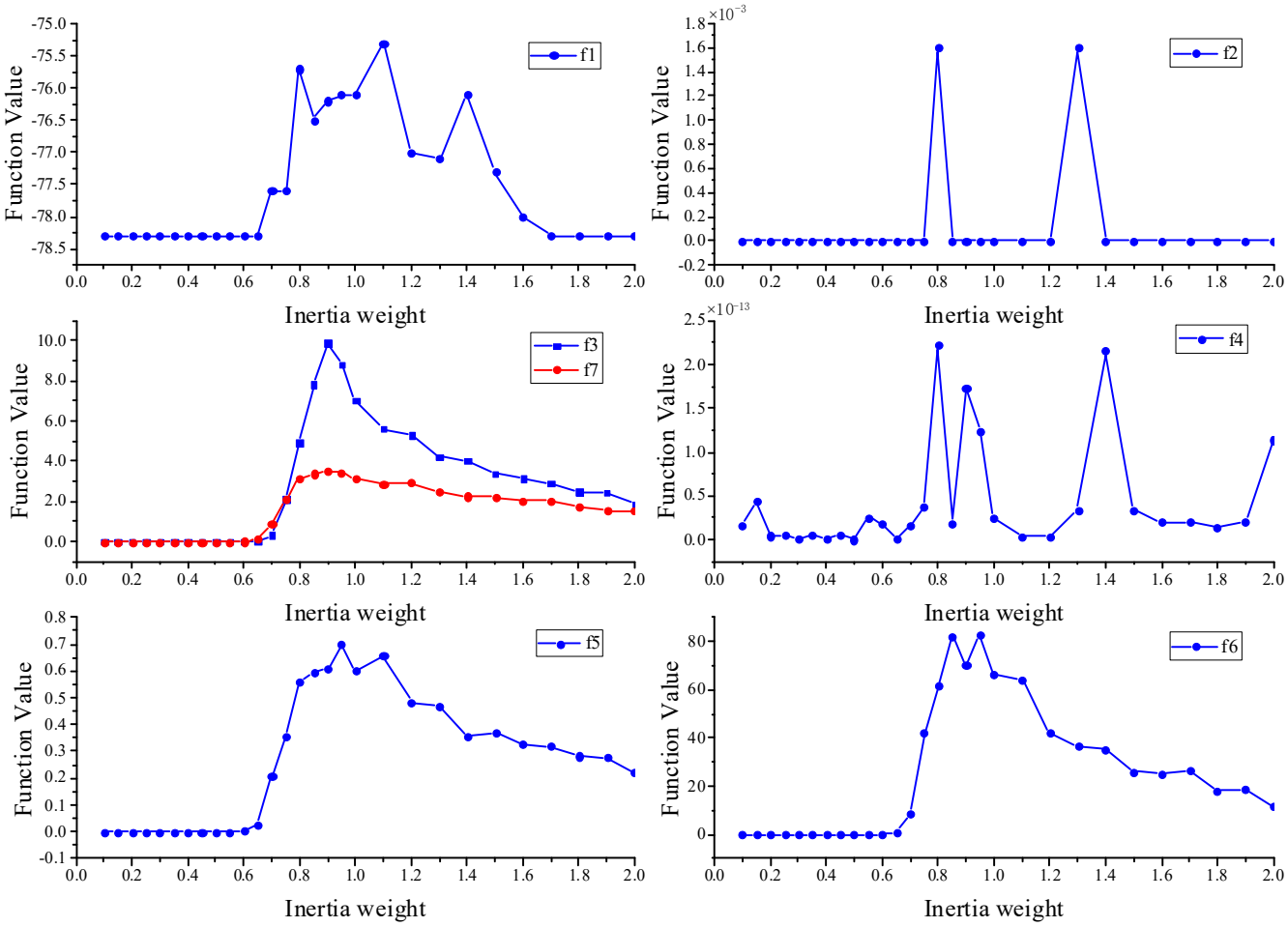

Figure 7. Inertia weight experiment.

Thus, it is reasonable to select ω from [0.1, 0.6] for SDPSO.

7.2. Parameter c_1 and c_2

For a reasonable range of parameter c_1 and c_2 for the SDPSO, given c_1, test the scope of c_2 in this experiment. When the percentage of success (the ratio of the number of successful runs to the total number of runs) is 100%, the value of c_2 can be noted. Results are filled into Tables 13 and 14 (the particle size is 100, $\omega = 0.3$, $\alpha = 3.0$, and $\beta = -0.5$).

Table 13. The scope of c_2 with different c_1 (c_1 = 0.1~1).

F	c_1 = 0.1	c_1 = 0.2	c_1 = 0.3	c_1 = 0.4	c_1 = 0.5	c_1 = 0.6	c_1 = 0.7	c_1 = 0.8	c_1 = 0.9	c_1 = 1
f1	2.4~4.0/40	2.6~4.0/40	2.5~4.0/30	2.5~4.0/30	2.2~3.9/20	0.9~3.8/70	1.1~3.8/50	1.0~3.4/90	1.0~3.2/90	1.1~3.2/60
f2	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1
f3	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1
f4	0.1~4.0/10	0.1~4.0/10	0.1~4.0/10	0.2~4.0/10	0.2~4.0/10	0.2~4.0/10	0.2~4.0/10	0.2~4.0/10	0.3~4.0/10	0.2~4.0/10
f5	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1
f6	0.2~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1
f7	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1

Note: (1) before the "/" is c_2, after that is the number of cycles when reach the optimal result; (2) operation time is relatively long when c_1 is greater than 2.0, and it is a difficult to gain an optimal solution.

Table 14. The scope of c_2 with different c_1 (c_1 = 1.1~2).

F	c_1 = 1.1	c_1 = 1.2	c_1 = 1.3	c_1 = 1.4	c_1 = 1.5	c_1 = 1.6	c_1 = 1.7	c_1 = 1.8	c_1 = 1.9	c_1 = 2
f1	1.1~3.1/90	1.0~2.7/50	1.0~2.8/80	1.1~2.7/80	1.1~2.5/100	1.1~2.3/120	1.1~2.1/180	1.1~2.1/210	1.0~2.0/430	1.0~2.0/370
f2	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~3.9/1	0.1~3.5/1
f3	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~3.9/1	0.1~3.8/1
f4	0.3~4.0/10	0.3~4.0/10	0.3~4.0/10	0.3~3.9/10	0.3~3.9/10	0.3~3.7/10	0.3~3.5/10	0.3~3.2/10	0.3~3.0/10	0.3~3.0/10
f5	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~3.9/1	0.1~3.8/1
f6	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1
f7	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~4.0/1	0.1~3.9/1	0.1~3.8/1	0.1~3.7/1

Note: (1) before the "/" is c_2, after that is the number of cycles when reach the optimal result; (2) operation time is relatively long when c_1 is greater than 2.0, and it is a difficult to gain an optimal solution.

From Tables 13 and 14, in most cases, when the acceleration factor $c_1 \in (0.1, 2.0)$ and $c_2 \in (0.1, 4.0)$, the success rate reaches 100%. The range of the parameters is so wide, indicating that the SDPSO is not sensitive to these parameters.

For these simple functions, we see that one cycle for most of them is enough to achieve the optimal value, which implies that SE and the DS (not PSO process) play main roles.

8. Conclusions

Although PSO has a good exploration ability and provides fast convergence, it suffers search stagnation from being trapped into a sub-optimal solution in an optimization. To promote local search ability of PSO, a novel hybrid algorithm (SDPSO) is proposed in this paper.

The SDPSO combines the SE (the search with the inertia-free velocity) and the DS (a component in the Rosenbrock method) with the original PSO. Thus, it maintains the global exploration ability of the PSO algorithm; meanwhile, the local search ability is reinforced by the union of the SE and the DS. For the SE, the weight inertia is cut from the velocity formula to balances the flight speed of a particle for reducing randomness. For the DS, moving near and along the ridge of a function enhance its local search ability.

In this paper, SDPSO is experimented with 30 unconstrained benchmark functions from CEC2014 and some constrained engineering design optimization problems. The performance of SDPSO is compared with that of other evolutionary algorithms and other PSO variants, and the results show that the SDPSO has overall good performance for the unconstrained benchmark functions and constrained problems.

At the end, the paper provides the parameter sensitivity analyses. The result shows that it is reasonable to select $\omega \in (0.1, 0.6)$, the acceleration factor $c_1 \in (0.1, 2.0)$ and $c_2 \in (0.1, 4.0)$, which can be a reference for the parameter selection of SDPSO for different problems. The parameter α and β of the DS affect the results, but we have not found their exact relations with the results. Future work will focus on a more comprehensive study of each parameter in the SDPSO.

Author Contributions: K.M. wrote the paper; K.M. conceived and designed the method; and Q.F. and W.K. performed the experiment tests. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 51478480.

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

Appendix A

Problem 1. *Pressure vessel design optimization problem*

Minimize:
$f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$
Subject to:
$g_1(X) = -x_1 + 0.0193x_3 \leq 0$
$g_2(X) = -x_2 + 0.00954 \leq 0$
$g_3(X) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0$
$g_4(X) = x_4 - 240 \leq 0$

where $X = (x_1, x_2, x_3, x_4)^T$. The ranges of the design parameters are $0 \leq x_1, x_2 \leq 99$, $10 \leq x_3, x_4 \leq 200$.

Problem 2. *Speed reducer design optimization problem*

Minimize:
$f(X) = 0.7854x_1x_2^2(3.3333x_2^3 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3)$
Subject to:
$g_1(X) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$
$g_2(X) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$
$g_3(X) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$
$g_4(X) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$
$g_5(X) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right)^{\frac{1}{2}}}{110.0x_6^3} - 1 \leq 0$
$g_6(X) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 157.5\times10^6\right)^{\frac{1}{2}}}{85.0x_7^3} - 1 \leq 0$
$g_7(X) = \frac{x_2x_3}{40} - 1 \leq 0$
$g_8(X) = \frac{5x_2}{x_1} - 1 \leq 0$
$g_9(X) = \frac{x_1}{12x_2} - 1 \leq 0$
$g_{10}(X) = \frac{1.5x_6+1.9}{x_4} - 1 \leq 0$
$g_{11}(X) = \frac{1.1x_7+1.9}{x_5} - 1 \leq 0$

where $X = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)^T$. The ranges of the design parameters are $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4, x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$ and $5.0 \leq x_7 \leq 5.5$.

Problem 3. *Spring design optimization problem*

Minimize:
$f(X) = (x_3 + 2)x_1x_2^2$
Subject to:
$g_1(X) = 1 - \frac{x_1^3x_3}{71,785x_2^4} \leq 0$
$g_2(X) = \frac{4x_1^2 - x_1x_2}{12,566(x_1x_2^3 - x_2^4)} + \frac{1}{5108x_2^2} - 1 \leq 0$
$g_3(X) = 1 - \frac{140.45x_2}{x_1^2x_3} \leq 0$
$g_4(X) = \frac{x_1+x_2}{1.5} - 1 \leq 0$

where $X = (x_1, x_2, x_3)^T$. The ranges of the design parameters are $0.25 \leq x_1 \leq 1.3$, $0.05 \leq x_2 \leq 2.0$ and $2 \leq x_3 \leq 15$.

Problem 4. *Welded beam design problem*

Minimize:
$f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$
Subject to:
$g_1(X) = \tau(X) - \tau_{\max} \leq 0$
$g_2(X) = \delta(X) - \delta_{\max} \leq 0$
$g_3(X) = \sigma(X) - \sigma_{\max} \leq 0$
$g_4(X) = x_1 - x_4 \leq 0$
$g_5(X) = P - P_c(X) \leq 0$
Parameters above are defined as follows:
$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2}$
$\tau' = \frac{P}{\sqrt{2}x_1x_2}$
$\tau'' = \frac{MR}{J}$
$M = P(L + \frac{x_2}{2})$
$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}$
$J = 2\left\{ \frac{x_1x_2}{\sqrt{2}} \left[\frac{x_2^2}{12} + (\frac{x_1+x_3}{2})^2 \right] \right\}$
$\sigma(X) = \frac{6PL}{x_4x_3^2}$
$\delta(X) = \frac{4PL^3}{Ex_4x_3^3}$
$P_c(X) = \frac{4.013\sqrt{EG(x_3^2x_4^6/36)}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$

where $X = (x_1, x_2, x_3, x_4)^T$, $P = 6000$ lb, $L = 14in$, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{\max} = 13,600$ psi, $\sigma_{\max} = 30,000$ psi and $\delta_{\max} = 0.25$ in. The ranges of the design parameters are $0.125 \leq x_1 \leq 2.0$, $0.1 \leq x_2, x_3 \leq 10.0$, $0.1 \leq x_4 \leq 2.0$.

Problem 5. *Three-bar truss design problem*

Minimize:
$f(X) = (2\sqrt{2}x_1 + x_2) \times l$
Subject to:
$g_1(X) = \frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P - \sigma \leq 0$
$g_2(X) = \frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P - \sigma \leq 0$
$g_3(X) = \frac{1}{x_1+\sqrt{2}x_2}P - \sigma \leq 0$

where $X = (x_1, x_2)^T$, $l = 100$ cm, $P = 2$ kN/cm^2 and $\sigma = 2$ kN/cm^2. The ranges of the design parameters are $0 \leq x_1, x_2 \leq 1$.

References

1. Wu, J.-Y. Stochastic Global Optimization Method for Solving Constrained Engineering Design Optimization Problems. In Proceedings of the 2012 Sixth International Conference on Genetic and Evolutionary Computing, Kitakyushu, Japan, 25–28 August 2012; pp. 404–408. [CrossRef]
2. Javaid, N.; Naseem, M.; Rasheed, M.B.; Mahmood, D.; Khan, S.A.; Alrajeh, N.; Iqbal, Z. A new heuristically optimized Home Energy Management controller for smart grid. *Sustain. Cities Soc.* **2017**, *34*, 211–227. [CrossRef]
3. Li, Z.; Zheng, X. Review of design optimization methods for turbomachinery aerodynamics. *Prog. Aerosp. Sci.* **2017**, *93*, 1–23. [CrossRef]
4. Li, X.; Tang, K.; Suganthan, P.; Yang, Z. Editorial for the special issue of Information Sciences Journal (ISJ) on "Nature-inspired algorithms for large scale global optimization". *Inf. Sci.* **2015**, *316*, 437–439. [CrossRef]
5. Boussaïd, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]
6. Jain, N.K.; Nangia, U.; Jain, J. A Review of Particle Swarm Optimization. *J. Inst. Eng. Ser. B* **2018**, *99*, 407–411. [CrossRef]

7. Vitorino, L.N.; Ribeiro, S.F.; Bastos-Filho, C.J.A. A mechanism based on Artificial Bee Colony to generate diversity in Particle Swarm Optimization. *Neurocomputing* **2015**, *148*, 39–45. [CrossRef]
8. Chen, Y.; Li, L.; Peng, H.; Xiao, J.; Yang, Y.; Shi, Y. Particle swarm optimizer with two differential mutation. *Appl. Soft Comput.* **2017**, *61*, 314–330. [CrossRef]
9. Jiao, B.; Lian, Z.; Gu, X. A dynamic inertia weight particle swarm optimization algorithm. *Chaos Solitons Fractals* **2008**, *37*, 698–705. [CrossRef]
10. Rosenbrock, H.H. An Automatic Method for Finding the Greatest or Least Value of a Function. *Comput. J.* **1960**, *3*, 175–184. [CrossRef]
11. Leader, J.J. *Numerical Analysis and Scientific Computation*; Pearson Addison Wesley: Boston, MA, USA, 2004; 590p.
12. Van den Berg, F.; Andrics, P.E. A Cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
13. 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]
14. Liang, J.; Suganthan, P. Dynamic multi-swarm particle swarm optimizer. In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, 2005 (SIS 2005), Pasadena, CA, USA, 8–10 June 2005; pp. 127–132. [CrossRef]
15. Sun, L.; Yoshida, S.; Cheng, X.; Liang, Y. A cooperative particle swarm optimizer with statistical variable interdependence learning. *Inf. Sci.* **2012**, *186*, 20–39. [CrossRef]
16. Li, J.; Zhang, J.; Jiang, C.; Zhou, M. Composite Particle Swarm Optimizer with Historical Memory for Function Optimization. *IEEE Trans. Cybern.* **2015**, *45*, 2350–2363. [CrossRef] [PubMed]
17. Gülcü, Ş.; Kodaz, H. A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization. *Eng. Appl. Artif. Intell.* **2015**, *45*, 33–45. [CrossRef]
18. Li, W.; Meng, X.; Huang, Y.; Fu, Z.-H. Multipopulation cooperative particle swarm optimization with a mixed mutation strategy. *Inf. Sci.* **2020**, *529*, 179–196. [CrossRef]
19. Xu, G.; Cui, Q.; Shi, X.; Ge, H.; Zhan, Z.-H.; Lee, H.P.; Liang, Y.; Tai, R.; Wu, C. Particle swarm optimization based on dimensional learning strategy. *Swarm Evol. Comput.* **2019**, *45*, 33–51. [CrossRef]
20. Li, W. Improving Particle Swarm Optimization Based on Neighborhood and Historical Memory for Training Multi-Layer Perceptron. *Information* **2018**, *9*, 16. [CrossRef]
21. He, S.; Wu, Q.; Wen, J.; Saunders, J.; Paton, R. A particle swarm optimizer with passive congregation. *Biosystems* **2004**, *78*, 135–147. [CrossRef] [PubMed]
22. Zeng, J.; Cui, Z.; Wang, L. A Differential Evolutionary Particle Swarm Optimization with Controller. *Lect. Notes Comput. Sci.* **2005**, *3612*, 467–476. [CrossRef]
23. Chen, K.; Zhou, F.; Yin, L.; Wang, S.; Wang, Y.; Wan, F. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Inf. Sci.* **2018**, *422*, 218–241. [CrossRef]
24. Liu, Y.; Qin, Z.; Xu, Z.-L.; He, X.-S. Using relaxation velocity update strategy to improve particle swarm optimization. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2005; Volume 4, pp. 2469–2472. [CrossRef]
25. Liu, Y.; Qin, Z.; He, X. Supervisor-student model in particle swarm optimization. In Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; Volume 1, pp. 542–547. [CrossRef]
26. Stacey, A.; Jancić, M.; Grundy, I. Particle swarm optimization with mutation. In Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, Australia, 8–12 December 2003; Volume 2, pp. 1425–1430. [CrossRef]
27. Miao, K.; Mao, X.; Li, C. Individualism of particles in particle swarm optimization. *Appl. Soft Comput.* **2019**, *83*, 105619. [CrossRef]
28. Miao, K.; Wang, Z. Neighbor-Induction and Population-Dispersion in Differential Evolution Algorithm. *IEEE Access* **2019**, *7*, 146358–146378. [CrossRef]
29. Tawhid, M.A.; Ali, A.F. Simplex particle swarm optimization with arithmetical crossover for solving global optimization problems. *OPSEARCH* **2016**, *53*, 705–740. [CrossRef]
30. Chen, Y.; Li, L.; Xiao, J.; Yang, Y.; Liang, J.; Li, T. Particle swarm optimizer with crossover operation. *Eng. Appl. Artif. Intell.* **2018**, *70*, 159–169. [CrossRef]
31. Parsopoulos, K.E.; Vrahatis, M.N. Recent approaches to global optimization problems through Particle Swarm Optimization. *Nat. Comput.* **2002**, *1*, 235–306. [CrossRef]
32. Zhang, W.-J.; Xie, X.-F. DEPSO: Hybrid particle swarm with differential evolution operator. In Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme—System Security and Assurance, Washington, DC, USA, 8 October 2004; Volume 4, pp. 3816–3821. [CrossRef]
33. Ibrahim, A.M.; Tawhid, M.A. A hybridization of cuckoo search and particle swarm optimization for solving nonlinear systems. *Evol. Intell.* **2019**, *12*, 541–561. [CrossRef]
34. Huang, C.-L.; Huang, W.-C.; Chang, H.-Y.; Yeh, Y.-C.; Tsai, C.-Y. Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering. *Appl. Soft Comput.* **2013**, *13*, 3864–3872. [CrossRef]
35. Javidrad, F.; Nazari, M.; Javidrad, H. Optimum stacking sequence design of laminates using a hybrid PSO-SA method. *Compos. Struct.* **2018**, *185*, 607–618. [CrossRef]

36. Luo, P.; Ni, P.; Yao, L.; Ho, S.; Ni, G.; Xia, H. Simulation of a new hybrid particle swarm optimization algorithm. *Int. J. Appl. Electromagn. Mech.* **2007**, *25*, 705–710. [CrossRef]
37. Wang, Y.-J.; Zhang, J.-S.; Zhang, Y.-F. A fast hybrid algorithm for global optimization. In Proceedings of the 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 August 2005; Volume 5, pp. 3030–3035. [CrossRef]
38. Salajegheh, F.; Salajegheh, E. PSOG: Enhanced particle swarm optimization by a unit vector of first and second order gradient directions. *Swarm Evol. Comput.* **2019**, *46*, 28–51. [CrossRef]
39. Hofmeister, B.; Bruns, M.; Rolfes, R. Finite element model updating using deterministic optimisation: A global pattern search approach. *Eng. Struct.* **2019**, *195*, 373–381. [CrossRef]
40. Bogani, C.; Gasparo, M.; Papini, A. Generalized Pattern Search methods for a class of nonsmooth optimization problems with structure. *J. Comput. Appl. Math.* **2009**, *229*, 283–293. [CrossRef]
41. Abramson, M.A.; Audet, C.; Chrissis, J.W.; Walston, J.G. Mesh adaptive direct search algorithms for mixed variable optimization. *Optim. Lett.* **2009**, *3*, 35–47. [CrossRef]
42. Liu, Y.; Qin, Z.; Shi, Z. Hybrid particle swarm optimizer with line search. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands, 10–13 October 2004; Volume 4, pp. 3751–3755. [CrossRef]
43. Fan, S.-K.S.; Liang, Y.-C.; Zahara, E. Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. *Eng. Optim.* **2004**, *36*, 401–418. [CrossRef]
44. El-Wakeel, A.S.; Smith, A.C. Hybrid Fuzzy-particle Swarm Optimization-simplex (F-PSO-S) Algorithm for Optimum Design of PM Drive Couplings. *Electr. Power Components Syst.* **2015**, *43*, 1560–1571. [CrossRef]
45. Kang, F.; Li, J.; Ma, Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Inf. Sci.* **2011**, *181*, 3508–3531. [CrossRef]
46. Lynn, N.; Suganthan, P.N. Ensemble particle swarm optimizer. *Appl. Soft Comput.* **2017**, *55*, 533–548. [CrossRef]
47. Palmer, J.R. An Improved Procedure for Orthogonalising the Search Vectors in Rosenbrock's and Swann's Direct Search Optimisation Methods. *Comput. J.* **1969**, *12*, 69–71. [CrossRef]
48. Lewis, R.M.; Torczon, V.; Trosset, M.W. Direct search methods: Then and now. *J. Comput. Appl. Math.* **2000**, *124*, 191–207. [CrossRef]
49. Sajid, I.; Ziavras, S.G.; Ahmed, M.M. FPGA-based normalization for modified gram-schmidt orthogonalization. In Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP 2010, Angers, France, 17–21 May 2010; Volume 2, pp. 227–232. [CrossRef]
50. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80. [CrossRef]
51. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
52. Fogel, L.J.; Owens, A.J.; Walsh, M.J. *Artificial Intelligence through Simulated Evolution*; Wiley: New York, NY, USA, 1966.
53. Akay, B.; Karaboga, D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *J. Intell. Manuf.* **2012**, *23*, 1001–1014. [CrossRef]
54. Simon, D. Biogeography-Based Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
55. Chen, H.G.; Wu, J.S.; Wang, J.L.; Chen, B. Mechanism study of simulated annealing algorithm. *Tongji Daxue Xuebao J. Tongji Univ.* **2004**, *32*, 802–805.
56. Zhan, Z.-H.; Zhang, J. Adaptive Particle Swarm Optimization. *Lect. Notes Comput. Sci.* **2008**, *5217*, 227–234. [CrossRef]
57. Zhan, Z.-H.; Zhang, J.; Li, Y.; Shi, Y.-H. Orthogonal Learning Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2011**, *15*, 832–847. [CrossRef]
58. Li, Y.-F.; Zhan, Z.-H.; Lin, Y.; Zhang, J. Comparisons study of APSO OLPSO and CLPSO on CEC2005 and CEC2014 test suits. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 3179–3185. [CrossRef]
59. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2014; pp. 1–32.
60. Aziz, N.A.A.; Ibrahim, Z.; Mubin, M.; Nawawi, S.W.; Mohamad, M.S. Improving particle swarm optimization via adaptive switching asynchronous–synchronous update. *Appl. Soft Comput.* **2018**, *72*, 298–311. [CrossRef]
61. Rada-Vilela, J.; Zhang, M.; Seah, W. A performance study on synchronicity and neighborhood size in particle swarm optimization. *Soft Comput.* **2013**, *17*, 1019–1030. [CrossRef]
62. Bonyadi, M.R.; Michalewicz, Z. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. *Evol. Comput.* **2017**, *25*, 1–54. [CrossRef]
63. Voglis, C.A.; Parsopoulos, K.E.; Lagaris, I.E. Particle swarm optimization with deliberate loss of information. *Soft Comput.* **2012**, *16*, 1373–1392. [CrossRef]
64. Chun, S.; Kim, Y.-T.; Kim, T.-H. A Diversity-Enhanced Constrained Particle Swarm Optimizer for Mixed Integer-Discrete-Continuous Engineering Design Problems. *Adv. Mech. Eng.* **2013**. [CrossRef]
65. Krohling, R.A.; Coelho, L.D.S. Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems. *IEEE Trans. Syst. Man Cybern. Part B* **2006**, *36*, 1407–1416. [CrossRef]

66. He, Q.; Wang, L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 1407–1422. [CrossRef]
67. Mazhoud, I.; Hadj-Hamou, K.; Bigeon, J.; Joyeux, P. Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1263–1273. [CrossRef]
68. Baykasoğlu, A.; Ozsoydan, F.B. Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Appl. Soft Comput.* **2015**, *36*, 152–164. [CrossRef]
69. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [CrossRef]
70. Savsani, P.; Savsani, V. Passing vehicle search (PVS): A novel metaheuristic algorithm. *Appl. Math. Model.* **2016**, *40*, 3951–3978. [CrossRef]
71. Montemurro, M.; Vincenti, A.; Vannucci, P. The Automatic Dynamic Penalisation method (ADP) for handling constraints with genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2013**, *256*, 70–87. [CrossRef]
72. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H.; Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Comput. Appl.* **2013**, *22*, 1239–1255. [CrossRef]
73. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
74. Zahara, E.; Kao, Y.-T. Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Syst. Appl.* **2009**, *36*, 3880–3886. [CrossRef]
75. Kiew, K.M.; Ray, T. Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior. *IEEE Trans. Evol. Comput.* **1903**, *7*, 386–396.
76. Wang, Y.; Cai, Z.; Zhou, Y. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. *Int. J. Numer. Methods Eng.* **2009**, *77*, 1501–1534. [CrossRef]
77. Wang, L.; Li, L.-P. An effective differential evolution with level comparison for constrained engineering design. *Struct. Multidiscip. Optim.* **2010**, *41*, 947–963. [CrossRef]
78. De Melo, V.V.; Carosio, G.L. Investigating Multi-View Differential Evolution for solving constrained engineering design problems. *Expert Syst. Appl.* **2013**, *40*, 3370–3377. [CrossRef]
79. Hwang, S.-F.; He, R.-S. A hybrid real-parameter genetic algorithm for function optimization. *Adv. Eng. Inform.* **2006**, *20*, 7–21. [CrossRef]
80. Zhang, M.; Luo, W.; Wang, X. Differential evolution with dynamic stochastic selection for constrained optimization. *Inf. Sci.* **2008**, *178*, 3043–3074. [CrossRef]
81. Zhang, J.; Liang, C.; Huang, Y.; Wu, J.; Yang, S. An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Appl. Math. Comput.* **2009**, *211*, 392–416. [CrossRef]
82. Shi, Y.; Eberhart, R.C. A Modified Particle Swarm. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 1945–1950.
83. Yang, C.; Simon, D. A New Particle Swarm Optimization Technique. In Proceedings of the 18th International Conference on Systems Engineering (ICSEng'05), Las Vegas, NV, USA, 16–18 August 2005; pp. 164–169. [CrossRef]
84. Eberhart, R.C.; Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In Proceedings of the 2000 Congress on Evolutionary Computation, CEC00, La Jolla, CA, USA, 16–19 July 2000; Volume 1, pp. 84–88. [CrossRef]