Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 175848, 12 pages
http://dx.doi.org/10.1155/2013/175848

Research Article

Improved Barebones Particle Swarm Optimization with Neighborhood Search and Its Application on Ship Design

Jingzheng Yao and Duanfeng Han

College of Shipbuilding Engineering, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Jingzheng Yao; yaojz_113@126.com

Received 30 August 2012; Revised 17 November 2012; Accepted 25 November 2012

Academic Editor: Baozhen Yao

Barebones particle swarm optimization (BPSO) is a new PSO variant, which has shown a good performance on many optimization problems. However, similar to the standard PSO, BPSO also suffers from premature convergence when solving complex optimization problems. In order to improve the performance of BPSO, this paper proposes a new BPSO variant called BPSO with neighborhood search (NSBPSO) to achieve a tradeoff between exploration and exploitation during the search process. Experiments are conducted on twelve benchmark functions and a real-world problem of ship design. Simulation results demonstrate that our approach outperforms the standard PSO, BPSO, and six other improved PSO algorithms.

1. Introduction

Particle swarm optimization (PSO), developed by Kennedy and Eberhart [1], is a new optimization technique inspired by swarm intelligence. Liker other evolutionary algorithms (EAs), PSO is also a population-based stochastic search algorithm, but it does not contain any crossover or mutation operator. During the search process, each particle adjusts its search behavior according to the search experiences of its previous best position (p_{best}) and the global best position (g_{best}). Due to its simplicity and easy implementation, PSO has been successfully applied to various practical optimization problems [2–5].

However, like other stochastic algorithms, PSO also suffers from premature convergence when handling complex multimodal problems. The main reason is that the attraction search pattern of PSO greatly depends on p_{best} and g_{best}. Once these best particles (p_{best} and g_{best}) get stuck, all particles in the swarm will quickly converge to the trapped position. In order to enhance the performance of PSO, different versions of PSO have been proposed in the past decades. Shi and Eberhart [6] introduced an inertia weight w into the original PSO to achieve a balance between the global and local search. Reported results show that a linearly decreased w is a parameter setting. Parsopoulos and Vrahaits [7] proposed a unified PSO (UPSO) which is a hybrid algorithm by combining the global and local versions of PSO. In [8], a fully informed PSO, called FIPS, is proposed by employing a modified velocity model. van den Bergh and Engelbrecht [9] used a cooperative mechanism to improve the performance of PSO on multimodal optimization problems. In the standard PSO, particles are attracted by their corresponding previous best particles and the global best particle. This search pattern is a greedy method which may result in premature convergence. To tackle this problem, Liang et al. [10] proposed a comprehensive learning PSO (CLPSO), in which each particle can be attracted by different previous best positions. Computational results on a set of multimodal problems demonstrate the effectiveness of CLPSO. In [11], Wang et al. proposed a new PSO algorithm (NSPSO) to search the neighbors of particles. This can provide more chances of finding better candidate solutions. Simulation studies show that NSPSO outperforms UPSO, FIPS, CPSO-H, and CLPSO. In [12], another version of NSPSO is proposed by employing neighborhood search and diversity enhanced mechanism.

Similar to other EAs, the performance of PSO also greatly depends on its control parameters, w, c_1, and c_2. The first parameter is known as inertia weight, and the last two are acceleration coefficients. Slight differences of these parameters may result in significantly different performance. To tackle this problem, some PSO variants based on adaptive parameters have been proposed to minimize the dependency of these parameters [13, 14]. Compared to these adaptive PSO algorithms, Kennedy developed a novel PSO called barebones PSO (BPSO) [15], which eliminates the velocity term and does not contain the parameters w, c_1, and c_2. In BPSO, a Gaussian sampling is used to generate new positions of particles. Empirical studies demonstrate that the performance of BPSO is competitive to the standard PSO and some improved PSO algorithms. Inspired by the idea of BPSO, some new algorithms haven been proposed. In [16], Omran et al. combined BPSO with differential evolution (DE) and the proposed barebones DE (BBDE). The reported results show that BBDE outperforms the standard DE and BPSO and it also achieves promising solutions for unsupervised image classification. Krohling and Mendel [17] introduced Gaussian and Cauchy mutations into BPSO to improve its performance. Experimental studies on a suite of well-known multimodal benchmark functions demonstrate the effectiveness of this approach. Blackwell [18] presented a theoretical analysis of BPSO. A series of experimental trials confirmed that the BPSO situated at the edge of collapse is comparable to other PSO algorithms and that performance can be still further improved with the use of an adaptive distribution. In [19], Wang embedded opposition-based learning (OBL) into BPSO to solve constrained nonlinear optimization problems. In addition, a new boundary search strategy is utilized. Simulation studies on thirteen constrained benchmark functions show that the new approach outperforms PSO, BPSO, and six other improved PSO algorithms.

In this paper, we also propose an improved barebones PSO called NSBPSO which employs a global and local neighborhood search strategies to make a balance between exploration and exploitation during the search process. In order to verify the performance of NSBPSO, twelve well-known benchmark functions and a real-world problem on ship design are used in the experiments. Computational results show that our approach outperforms PSO, BPSO, and several other improved PSO variants in terms of the quality of solutions.

The rest of the paper is organized as follows. The standard PSO and barebones PSO are given in Section 2. In Section 3, our approach NSBPSO is proposed. Experimental studies are presented in Section 4. Section 5 presents a real-world application on ship design. Finally, the work is summarized in Section 6.

2. Barebones Particle Swarm Optimization

PSO is a population-based stochastic search algorithm, which simulates the behaviors of fish schooling or birds flocking. Each particle has a velocity and a position vectors. During the search space, a particle dynamically adjusts its velocity to generate a new position as follows:

$$\begin{aligned} V_i(t+1) &= w \cdot V_i(t) + c_1 \cdot r_1 \cdot \left(p_{\text{best}_i} - X_i(t)\right) \\ &\quad + c_2 \cdot r_2 \cdot \left(g_{\text{best}} - X_i(t)\right), \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \tag{1}$$

where X_i and V_i are the position and velocity vector for the ith particle, respectively. p_{best_i} is the previous best particle of the ith particle and g_{best} is the global best particle. r_1 and r_2 are two independently generated random numbers within $[0, 1]$. The parameter w is known as inertia weight. c_1 and c_2 are acceleration coefficients.

A recent study [20] proved that the particles in PSO converge to the weighted position of p_{best} and g_{best} as follows:

$$\lim_{t \to +\infty} X_i(t) = \frac{c_1 \cdot p_{\text{best}_i} + c_2 \cdot g_{\text{best}}}{c_1 + c_2}. \tag{2}$$

Based on the convergence characteristic of PSO, Kennedy [15] proposed a new PSO variant called barebones PSO (BPSO), in which each particle only has a position vector and eliminates the velocity vector. Therefore, BPSO does not contain the parameters w, c_1, and c_2. In BPSO, a new position is updated by Gaussian sampling as follows:

$$X_i(t+1) = N\left(\frac{g_{\text{best}} + p_{\text{best}_i}}{2}, \left|g_{\text{best}} - p_{\text{best}_i}\right|\right), \tag{3}$$

where $N(\cdot)$ indicates a Gaussian distribution with mean $(g_{\text{best}} + p_{\text{best}_i})/2$ and standard deviation $|g_{\text{best}} - p_{\text{best}_i}|$.

3. Barebones PSO with Neighborhood Search

Due to the intrinsic randomness, both PSO and EAs suffer from premature convergence when solving complex multimodal problems. Sometimes, the suboptima are near to the global optimum and the neighborhoods of trapped particles may contain the global optimum. For this case, searching the neighbors of particles is beneficial for finding better solutions. Based on this idea, some neighborhood search strategies have been successfully applied to various algorithms.

In [11], Wang et al. proposed a new PSO algorithm called PSO with neighborhood search strategies (NSPSO), which utilizes one local and two global neighborhood search strategies. The NSPSO includes two operations. First, for each particle, three trial particles are generated by the above neighborhood search strategies, respectively. Second, the best one among the three trial particles and the current particle is chosen as the new current particle. Simulation studies on twelve unimodal and multimodal benchmark problems show that NSPSO achieves better results than standard PSO and five other improved PSO algorithms.

Although NSPSO has shown good search abilities, its performance is still seriously influenced by its control parameters, w, c_1, and c_2. In [11], NSPSO used an empirical parameter settings, $c_1 = c_2 = 1.49618$ and $w = 0.72984$. In order to minimize the effects of the control parameters

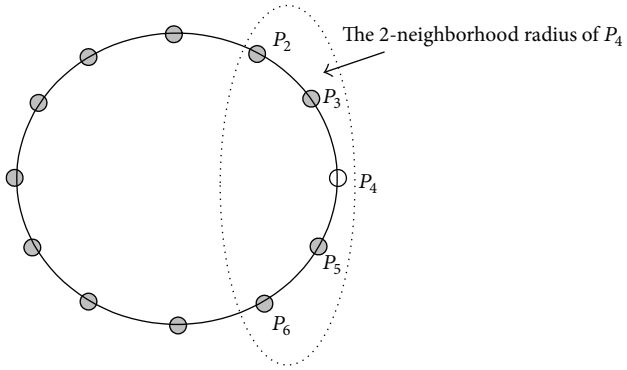

Figure 1: The ring topology and 2-neighborhood radius.

on the performance of NSPSO, this paper proposes an improved PSO algorithm by combining barebones PSO and the neighborhood search strategies.

There are various population topologies, such as ring, wheel, star, Von Neumann, and random. A recent study shows that the complexity of population topology affects the performance of PSO. A population topology with few connections (low complexity) may perform well on multimodal problems, while a highly interconnected population topology may perform well on unimodal problems. In this paper, a ring topology is used by the suggestions of [11].

The ring topology assumes that particles are organized as a ring. In [21], a special ring topology is proposed by connecting the indices of particles. For example, the fourth particle P_4 is connected by the third one P_3 and the fifth one P_5. In other words, P_3 and P_5 are two immediate neighbors of P_4. Figure 1 shows the employed ring topology. Based on the ring topology, a k-neighborhood radius is defined, where k is a predefined integer number. For each particle P_i, its k-neighborhood radius consists $2k+1$ particles (include itself), which are $P_{i-k}, \ldots, P_{i-1}, P_i, P_{i+1}, \ldots, P_{i+k}$. It is obvious that the parameter k satisfies $0 \le k \le (N-1)/2$. Figure 1 shows the 2-neighborhood radius of P_4, where 5 particles are covered by the neighborhood. By the suggestions of [11], $k = 2$ is used in this paper.

Based on the k-neighborhood radius, a local neighborhood search strategy is proposed. For each particle P_i, a local particle L_i is generated as follows [11]:

$$LX_i(t+1) = a_1 \cdot X_i(t) + a_2 \cdot p_{\text{best}_i} + a_3 \cdot \left(X_{i1}(t) - X_{i2}(t)\right), \tag{4}$$

where X_{i1} and X_{i2} are the position vectors of two particles, P_{i1} and P_{i2}, randomly selected from the k-neighborhood neighborhood, $i1, i2 \in [i-k, i+k] \wedge i1 \ne i2 \ne i$, a_1, a_2, a_3 are three random numbers within $(0, 1)$, and $a_1 + a_2 + a_3 = 1$. In [11], the velocity of L_i keeps the same with P_i. Although the velocity mechanism is simple, it may not beneficial for the next flight of L_i. Therefore, we use a similar method to generate LV_i:

$$LV_i(t+1) = a_1 \cdot V_i(t) + a_2 \cdot p_{\text{best}_{V_i}} + a_3 \cdot \left(V_{i1}(t) - V_{i2}(t)\right), \tag{5}$$

where $p_{\text{best}_{V_i}}$ is the velocity vector of p_{best_i} and V_{i1}, V_{i2} are the velocity vectors of P_{i1} and P_{i2}, respectively.

Beside the local neighbor strategy, a global neighborhood search strategy is proposed. For each particle P_i, a global particle G_i is generated as follows [11]:

$$GX_i(t+1) = a_1 \cdot X_i(t) + a_2 \cdot g_{\text{best}} + a_3 \cdot \left(X_{i3}(t) - X_{i4}(t)\right), \tag{6}$$

where X_{i3} and X_{i4} are the position vectors of two particles, P_{i3} and P_{i4}, randomly selected from the current swarm, $i3, i4 \in [1, N] \wedge i3 \ne i4 \ne i$, a_1, a_2, a_3 are three random numbers within $(0, 1)$, and $a_1 + a_2 + a_3 = 1$. In [11], the velocities of G_i keeps the same with P_i. So, the velocity of L_i, G_i, and P_i are the same, but L_i (local) and G_i (global) are two different types of particles. Like (5), this paper uses a new method to generate GV_i:

$$LV_i(t+1) = a_1 \cdot V_i(t) + a_2 \cdot g_{\text{best}_V} + a_3 \cdot \left(V_{i3}(t) - V_{i4}(t)\right), \tag{7}$$

where g_{best_V} is the velocity vector of g_{best} and V_{i3}, V_{i4} are the velocity vectors of P_{i3} and P_{i4}, respectively.

After generating two new particles L_i and G_i, a greedy selection mechanism is used. Among P_i, L_i, and G_i, we select the best one as the new P_i.

In our approach NSBPSO, it embeds the local and global neighborhood search strategies into barebones PSO. The neighborhood search strategies focus on searching the neighbors of particles and provide different search behaviors. The BPSO concentrates on minimizing the dependency of the control parameters (without w, c_1, and c_2). By hybridization of BPSO and the neighborhood strategies, NSBPSO is almost a parameter-free algorithm (except for the probability of the neighborhood search), which achieves a tradeoff between exploration and exploitation.

The main steps of NSBPSO are listed as follows.

Step 1. Randomly initialize the swarm, and evaluate the fitness values of all particles.

Step 2. Initialize p_{best} and g_{best}.

Step 3. For each particle P_i, calculate its new position vector X_i according to (3). Evaluate the fitness value of P_i. If needed, update p_{best_i} and g_{best}.

Step 4. For each particle P_i, if r and $(0, 1) < P_{\text{ns}}$, where r and $(0, 1)$ is a random number within $[0, 1]$ and P_{ns} is the probability of conducting neighborhood search, then go to Step 5; otherwise go to Step 6.

Step 5. Generate a local particle L_i according to (4) and (5). Generate a local particle G_i according to (6) and (7). Evaluate the fitness values of L_i and G_i. Among P_i, L_i, and G_i, we select the best one as the new P_i. If needed, update p_{best_i} and g_{best}.

Step 6. If the stop criterion is satisfied, then stop the algorithm and output the results; otherwise go to Step 3.

Table 1: The twelve benchmark problems.

Problems	D	Search range
$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]$
$f_2(x) = \sum_{i=1}^{D-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$	30	$[-2.048, 2.048]$
$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + e$	30	$[-32.768, 32.768]$
$f_4(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$
$f_5(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\cdot 0.5\right)\right]$ $a = 0.5, b = 3, k\max = 20$	30	$[-0.5, 0.5]$
$f_6(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	30	$[-5.12, 5.12]$
$f_7(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos\left(2\pi y_i\right) + 10\right)$ $y_i = \begin{cases} x_i, & \lvert x_i\rvert < \frac{1}{2} \\ \frac{\text{round}\left(2x_i\right)}{2}, & \lvert x_i\rvert \geq \frac{1}{2} \end{cases}$	30	$[-5.12, 5.12]$
$f_8(x) = 418.9829\cdot D - \sum_{i=1}^{D}\left(x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500, 500]$
$f_9(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi y_i\right)\right) + 20 + e$ $y = \mathbf{M} * x$	30	$[-32.768, 32.768]$
$f_{10}(x) = \sum_{i=1}^{D}\frac{y_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{y_i}{\sqrt{i}}\right) + 1$ $y = \mathbf{M} * x$	30	$[-600, 600]$
$f_{11}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(y_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\cdot 0.5\right)\right]$ $y = \mathbf{M} * x$	30	$[-0.5, 0.5]$
$f_{12}(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos\left(2\pi y_i\right) + 10\right)$ $y = \mathbf{M} * x$	30	$[-5.12,\ 5.12]$

4. Experimental Study

4.1. Test Problems. In order to verify the performance of our approach, there are twelve well-known benchmark problems used in the following experiments [10]. According to the properties of these problems, they are divided into two three types: unimodal problems (f_1–f_2), unrotated multimodal problems (f_3–f_8), and rotated multimodal problems (f_9–f_{12}). For rotated problems, the original variable x is left multiplied by the orthogonal matrix $\mathbf{M}$ to get the new rotated variable $y = \mathbf{M} * x$. For all test problems, they are to be minimized and their global optima are zero. The specific descriptions of these problems are presented in Table 1.

4.2. Effects of the Parameter P_{ns}. The main contribution of this paper is to minimize the effects of the control parameters and improve the performance of BPSO. Although NSBPSO eliminates the control parameters, w, c_1, and c_2, it introduces two new parameters k and P_{ns}. The parameter k is the size of neighborhood radius. The ring population topology used in this paper assumes that particles are connected by their indices. Although P_2 and P_3 are two neighbors of P_1, they may not be the nearest one to P_1 (Euclidean distance). So, the size of the neighborhood radius does not affect the selection of particles in the local neighborhood search. Our empirical studies also confirm it (here we do not list the results of NSBSO with different k-neighborhood radius). According to the suggestions of [11], k is set to 2 in this paper.

Table 2: Results achieved by NSBPSO with different P_{ns}.

Problems	$P_{ns} = 0.0$ Mean	$P_{ns} = 0.1$ Mean	$P_{ns} = 0.3$ Mean	$P_{ns} = 0.5$ Mean	$P_{ns} = 0.7$ Mean	$P_{ns} = 1.0$ Mean
f_1	$3.60E-87$	$1.47E-285$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_2	$1.84E+01$	**1.81E + 01**	$2.01E+01$	$2.08E+01$	$2.13E+01$	$2.23E+01$
f_3	$1.12E-14$	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**
f_4	$9.86E-03$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_5	$5.65E-05$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_6	$3.38E+01$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_7	$2.50E+01$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_8	$2.13E+03$	$1.80E+03$	**1.01E + 03**	$2.78E+03$	$2.63E+03$	$4.17E+03$
f_9	$1.90E+00$	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**	**5.89E − 16**
f_{10}	$2.95E-02$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_{11}	$7.95E+00$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_{12}	$6.07E+01$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**

The parameter P_{ns} controls the probability of conducting neighborhood search. A larger P_{ns} will result in more neighborhood search operations, while a smaller P_{ns} will have less. This may affect the performance of NSBPSO. To investigate the effects of P_{ns}, this section presents an experimental study. In the experiment, the P_{ns} is set to 0.0, 0.1, 0.3, 0.5, 0.7, and 1.0, respectively. The performance of NSBPSO with different P_{ns} is compared.

For other parameters of NSBPSO, we use the following settings by the suggestions of [10]. The population size N is set to 40. When the number of fitness evaluations (FEs) reaches the maximum value MAX_FEs, the algorithm stops running. In the experiment, MAX_FEs is set to $2.0e+05$. For each test problem, NSBPSO is run 30 times and the mean fitness error values are reported.

Table 2 presents the computational results of NSBPSO under different P_{ns}, where "Mean" represents the mean fitness error values. The best results among the comparison are shown in boldface. As seen, the performance of NSBPSO is not sensitive to the parameter P_{ns}. A smaller ($P_{ns} < 0.3$) or larger ($P_{ns} > 0.5$) value of P_{ns} almost achieves similar results. For $P_{ns} = 0.0$, NSBPSO is equal to the original BPSO, because the neighborhood search operations are not conducted. For this case, the algorithm shows poor performance and falls into local minima on most test functions. When $P_{ns} = 0.1$, NSBPSO significantly outperforms NSBPSO with $P_{ns} = 0.0$. It demonstrates that the neighborhood search strategies are very effective. Even if we use a small P_{ns}, NSBPSO can also obtain promising results.

The value of P_{ns} does not affect the performance of NSBPSO, and $P_{ns} > 0$ is applicable for all test problems. In this paper, $P_{ns} = 0.3$ is used in the following experiments.

Figure 2 presents the convergence processes of NSBPSO with different P_{ns}. Although different P_{ns} of NSBPSO can find the global optimum on the majority of test functions, they show different convergence characteristics. For problem f_1, larger P_{ns} converges faster than smaller P_{ns}. For problem f_2, $P_{ns} = 0.1$ converges fastest than other values. For f_8, $P_{ns} = 0.3$ shows the fastest convergence speed.

4.3. Comparison of NSBPSO with Other PSO Algorithms. In this section, experiments are conducted to compare nine PSO algorithms including the proposed NSBPSO on the 12 test problems. The involved algorithms are listed as follows.

(1) standard PSO,

(2) barebones PSO (BPSO),

(3) unified PSO (UPSO) [7],

(4) fully informed PSO (FIPS) [8],

(5) cooperative PSO (CPSO-H) [9],

(6) comprehensive learning PSO (CLPSO) [10],

(7) adaptive learning PSO (APSO) [13],

(8) pSO with neighborhood search (NSPSO) [11],

(9) our approach (NSBPSO).

For the sake of fair comparison, we use the same settings for the same parameters. For all algorithms, the population size N is set to 40, and the maximum number of fitness evaluations (MAX_FEs) is set to $2.0e+05$. For standard PSO, w is linearly decreased from 0.9 to 0.4, and $c_1 = c_2 = 1.49618$. For NSPSO, the probability of neighborhood search P_{ns} is set to 0.3. The parameter settings of UPSO, CPSO-H, FIPS, and CLPSO are described in [10]. For NSPSO and APSO, the same parameters are used by the literature [11, 13], respectively. For each test problem, each algorithm is conducted 30 times and the mean fitness error values are reported.

Table 3 lists the comparison results of NSBPSO with other eight PSO algorithms, where "Mean" represents the mean fitness error values. The best results are shown in boldface. From the results, it can be seen that NSBPSO outperforms PSO, BPSO, and FIPS on all test problems. UPSO and APSO perform better than NSBPSO on f_2, while NSBPSO achieves better results for the rest 11 problems. CPSO-H obtains better solution than NSBPSO on f_2, while NSBPSO outperforms CPSO-H on 10 problems. For problem f_6, both NSBPSO and CPSO-H can find the global optimum. NSPSO performs

Table 3: Results achieved by the nine PSO algorithms.

Algorithms	f_1 Mean	f_2 Mean	f_3 Mean	f_4 Mean	f_5 Mean	f_6 Mean	f_7 Mean	f_8 Mean	f_9 Mean	f_{10} Mean	f_{11} Mean	f_{12} Mean
PSO	$9.78E-30$	$2.93E+01$	$3.94E-14$	$8.13E-03$	$1.30E-04$	$2.90E+01$	$2.97E+01$	$1.10E+03$	$2.80E-01$	$1.64E-01$	$6.66E-01$	$9.90E+00$
BPSO	$3.60E-87$	$1.84E+01$	$1.12E-14$	$9.86E-03$	$5.65E-05$	$3.38E+01$	$2.50E+01$	$2.13E+03$	$1.90E+00$	$2.95E-02$	$7.95E+00$	$6.07E+01$
UPSO	$4.17E-87$	$1.51E+01$	$1.22E-15$	$1.66E-03$	$9.60E+00$	$6.59E+01$	$6.34E+01$	$4.84E+03$	$1.00E+00$	$7.76E-02$	$2.61E+00$	$1.52E+01$
FIPS	$2.69E-12$	$2.45E+01$	$4.81E-07$	$1.16E-06$	$1.54E-01$	$7.30E+01$	$6.08E+01$	$2.05E+03$	$2.25E-15$	$1.70E-01$	$5.93E-14$	$1.20E+01$
CPSO-H	$1.16E-113$	$7.08E+00$	$4.93E-14$	$3.63E-02$	$7.82E-15$	**$0.00E+00$**	$1.00E-01$	$1.08E+03$	$1.36E+00$	$1.20E-01$	$4.35E+00$	$2.67E+01$
CLPSO	$1.46E-14$	$2.01E+01$	**$0.00E+00$**	$3.14E-10$	$3.45E-07$	$4.85E-10$	$4.36E-10$	**$1.27E-12$**	$3.65E-05$	$4.50E-02$	$3.72E-10$	$5.97E+00$
APSO	$9.60E-66$	$1.83E+01$	$1.09E-14$	$1.20E-02$	$4.77E-02$	$6.27E+00$	$2.27E+00$	$2.13E+03$	$1.22E+00$	$1.38E-02$	$8.40E+00$	$7.09E+01$
NSPSO	**$0.00E+00$**	**$4.08E+00$**	$5.89E-16$	**$0.00E+00$**	$6.72E-13$	**$0.00E+00$**	**$0.00E+00$**	$1.07E+03$	$9.55E-13$	**$0.00E+00$**	$1.54E-13$	**$0.00E+00$**
NSBPSO	**$0.00E+00$**	$2.01E+01$	$5.89E-16$	**$0.00E+00$**	**$0.00E+00$**	**$0.00E+00$**	**$0.00E+00$**	$1.01E+03$	**$5.89E-16$**	**$0.00E+00$**	**$0.00E+00$**	**$0.00E+00$**

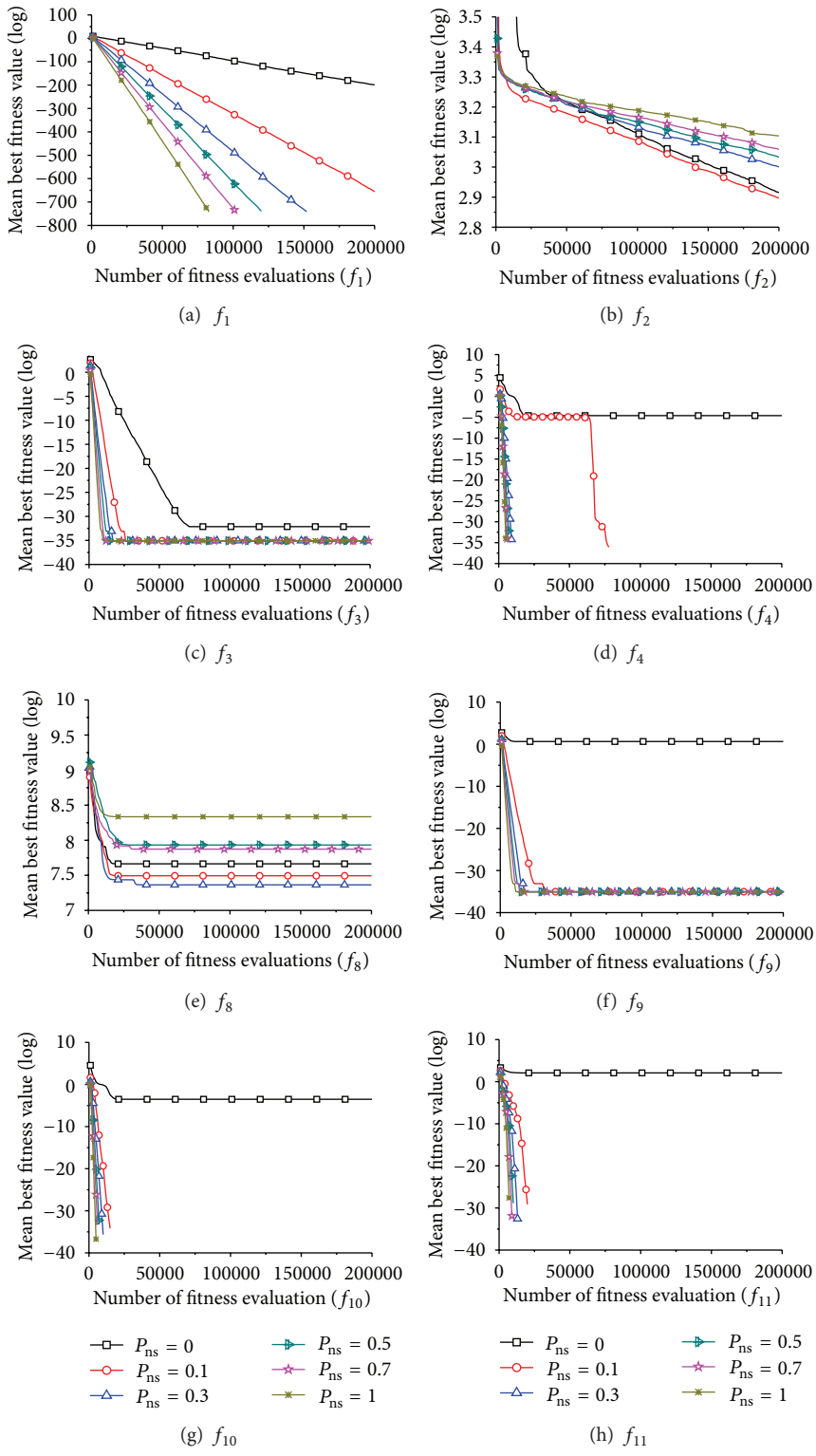

Figure 2: The convergence curves of NSBPSO with different P_{ns}.

Table 4: Results of Wilcoxon signed-rank test between NSBPSO and other eight PSO algorithms.

NSBPSO versus	P values
PSO	**4.88E − 04**
BPSO	**6.05E − 03**
UPSO	**6.35E − 03**
FIPS	**1.17E − 02**
CPSO-H	**4.88E − 04**
CLPSO	$6.54E-02$
APSO	**6.35E − 03**
NSPSO	$3.75E-01$

better than NSBPSO on f_2, while NSBPSO outperforms NSPSO on 4 problems. Both of them can converge to the global optimum on 7 problems.

From the comparison of BPSO and PSO, BPSO outperforms PSO on 6 problems, while PSO achieves better results than BPSO for the rest 6 problems. The results demonstrate that the performance of BPSO is similar to PSO on these problems. Compared to PSO, BPSO is more competitive, because BPSO does not contain any control parameter (except for the population size), while PSO employs empirical parameter settings. By hybridization of BPSO (or PSO) and the neighborhood search, NSBPSO (or NSPSO) achieves significantly improvements on the performance of BPSO (or PSO). Compared to NSPSO, NSBPSO not only achieves better results, but also has less control parameters.

In order to compare the performance differences between NSBPSO and the other eight PSO algorithms, we conduct the Wilcoxon signed-rank test by the suggestions of [22]. Table 4 shows the P-values achieved by the Wilcoxon test. The P values below 0.05 are shown in boldface. As shown, NSBPSO is significantly better than all other algorithms except for CLPSO and NSPSO. Though NSBPSO is not significantly better than them, it outperforms them in the majority of test problems.

5. Application on Ship Design

5.1. Problem Description. This section investigates the performance of our approach NSBPSO for a conceptual ship design. The original optimization statements are presented in [23, 24]. The ship design optimization problem used in this paper has six design variables, three objectives, and 9 inequality constraints. The design variables are length (L), beam (B), depth (D), draft (T), block coefficient (C_B), and speed in knots (V_k). The ship design problem aims to minimize transportation cost (T_c) and lightship weight (L_s) and maximize annual cargo (A_c) [24]:

$$\begin{aligned} &\text{Minimize} \quad \{T_c, L_s\} \\ &\text{Maximize} \quad A_c, \end{aligned} \tag{8}$$

where $T_c = C_A/A_c$, $A_c = (\text{DWT} - F_c - \text{DWT}_M) \cdot \text{RTPA}$, and $L_s = W_s + W_o + W_m$. The specific model definition is described in Table 5 [25]. The search ranges of the six variables L, B, D, T, C_B, and V_k are listed in Table 6.

There are 9 inequality constraints listed as follows:

$$\begin{gathered} 6 - \frac{L}{B} \le 0, \\ \frac{L}{D} - 15 \le 0, \\ \frac{L}{T} - 19 \le 0, \\ T - 0.45\text{DWT}^{0.31} \le 0, \\ T - 0.7D + 0.7 \le 0, \\ \text{DWT} - 500000 \le 0, \\ 25000 - \text{DWT} \le 0, \\ F_n - 0.32 \le 0, \\ 0.07B - KB - BMT + KG \le 0. \end{gathered} \tag{9}$$

5.2. Constraint Handling. In order to deal with the constraints, an adaptive penalty method is employed by the suggestions of [19]. Let $f(x)$ be the objective function (T_c, L_s, or A_c). The fitness evaluation function $F(x)$ is defined by

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ \overline{f}(x) + \sum_{j=1}^{m} k_j \cdot CV_j, & \text{otherwise,} \end{cases} \tag{10}$$

where m is the number of inequality constraints, $CV_j = \min\{0, c_j(x)\}$ is the constraint violation for the jth constraint, $c_j(x)$ is the jth constraint, $\overline{f}(x)$ is the mean objective function value in the current swarm, and k_j is a penalty coefficient defined as follows:

$$k_j = \frac{\left|f(x)\right| \cdot \overline{CV_j}(x)}{\sum_{i=1}^{m} \left[\overline{CV_i}(x)\right]^2}, \tag{11}$$

where $\overline{CV_j}(x)$ is the average violation of the jth constraint for all particles in the swarm.

5.3. Computational Results. The ship design problem is a multiobjective optimization problem which has three objectives. By the suggestions of [24, 25], this paper only considers single objective optimization. Therefore, the whole problem is divided into three single objective optimization problems: (1) minimize transportation cost (T_c), (2) minimize lightship weight (L_s), and (3) maximize annual cargo (A_c).

In this section, we conduct three series of experiments for the three single optimization problems. In order to verify the performance of our approach NSBPSO, we compare it with four other algorithms. The involved algorithms are listed as follows:

(1) parsons and Scott's method [24],

(2) standard PSO,

Table 5: Model definition of the ship design problem.

Parameter	Definition	Parameter	Definition
Steel weight (W_s)	$0.034\,L^{1.7}B^{0.7}D^{0.4}C_B^{0.5}$	Port cost (C_P)	$6.3\,\mathrm{DWT}^{0.8}$
Outfit weight (W_o)	$1.0\,L^{0.8}B^{0.6}D^{0.4}C_B^{0.5}$	Fuel carried (F_c)	$D_c(D_s + 0.5)$
Displacement (Δ)	$1.025LBTC_B$	Misc. deadweight (DWT_M)	$2.0\,\mathrm{DWT}^{0.5}$
Machinery weight (W_m)	$0.17\,P^{0.9}$	Cargo deadweight (DWT_c)	$\mathrm{DWT}-F_c-\mathrm{DWT}_M$
Capital cost (C_c)	$0.2\,C_s$	Port days (D_p)	$2\,(\mathrm{DWT}_c/H_R + 0.5)$
Running cost (C_R)	$40000\,\mathrm{DWT}^{0.3}$	Round trips per year (RTPA)	$350/(D_s + D_p)$
Daily cost (D_c)	$(0.19 \cdot 24 \cdot P/1000) + 0.2$	Voyage cost (C_V)	$(C_F + C_P)$ RTPA
Handling rate (H_R)	8000 (t/day)	Round trip miles (RTM)	5000 (nm)
Fuel price (F_P)	100 (£/t)	Sea days (D_s)	$\mathrm{RTM}/24\,V_k$
Power (P)	$\sqrt[3]{\Delta^2 V_k^3}/(a + bF_n)$	Fuel cost (C_F)	$1.05\,D_C D_S F_P$
Froude number (F_n)	$V/\sqrt{gL}, V = 0.5144V_k$	g	9.8065
Annual cost (C_A)	(DWT_C) RTPA	KB	$0.53\,T$
KG	$1.0 + 0.52D$	BMT	$(0.085\,C_B - 0.002)B^2/(TC_B)$
Ship cost (C_s)		$1.3\,(2000\,W_s^{0.85} + 3500\,W_o + 2400\,P^{0.8})$	

Table 6: Search ranges of the six design variables.

$150 \le L \le 274.32$	$10 \le T \le 11.71$
$20 \le B \le 32.31$	$0.63 \le C_B \le 0.75$
$13 \le D \le 25$	$11 \le V_k \le 20$

(3) barebones PSO (BPSO),

(4) PSO with neighborhood search (NSPSO),

(5) our approach (NSBPSO).

To have a fair comparison, the same parameter settings are used for common parameters. For all algorithms, the population size and the maximum number of fitness evaluations (MAX_FEs) are set to 100 and 1.0e+06. For standard PSO, w is linearly decreased from 0.9 to 0.4 and $c_1 = c_2 = 1.49618$. For NSBPSO and NSPSO, P_{ns} is set to 0.3. For each optimization problem, each algorithm is run 10 times and the best results among these runs are presented.

Tables 7–9 show the computational results for the three problems. For Table 7, NSBPSO achieves the minimal transportation cost among the five algorithms; but it also obtains the minimal value of annual cargo. For the objective of T_c, NSBPSO is the best among the five algorithms, however, it could not obtain the best results for all three objectives. For annual cargo A_c, Parsons and Scott's [24] method is the best. Tables 8 and 9 can also get similar conclusions. The results demonstrate that NSBPSO shows better performance than the other three algorithms for single objective optimization problem of the ship design. When considering all objectives, we cannot conclude which algorithm is the best. To perfectly solve this problem, we may use multiobjective optimization algorithms. Figures 3, 4, and 5 present the convergence curves of PSO, BPSO, NSPSO, and NSBPSO for the three single objective problems. For minimizing transportation cost, NSBPSO shows faster convergence speed at the last stage of the evolution. For minimizing lightship weight, NSBPSO converges faster than other three PSO algorithms.

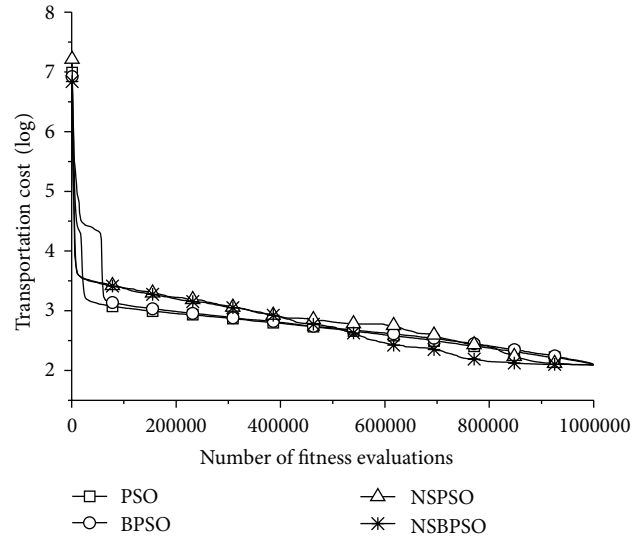

Figure 3: The convergence curves of PSO, BPSO, NSPSO and NSBPSO for minimizing transportation cost.

For maximizing annual cargo, both NSBPSO and NSPSO show similar convergence characteristics.

6. Conclusions

Barebones PSO (BPSO) is a new variant of PSO which eliminates the velocity term. Although some reported results show that BPSO is better than PSO, it still gets stuck when solving complex multimodal problems. In order to enhance the performance of BPSO, this paper proposes an improved version called BPSO with neighborhood search (NSBPSO). The new approach embeds one local and one global neighborhood search strategies into the original BPSO to achieve a tradeoff between exploration and exploitation. Compared to other improved PSO algorithms, NSBPSO is

Table 7: Computational results for minimizing transportation cost.

Objective	Parsons and Scott [24]	PSO	BPSO	NSPSO	NSBPSO
Transportation cost (T_c)	8.377	8.213	8.195	8.176	8.162
Lightship weight (L_s)	9029.0	8420.1	8486.3	8452.8	8355.2
Annual cargo (A_c)	551265	509078	508069	505738	503261
L	193.86	190.63	191.72	193.04	192.36
B	32.31	30.15	30.13	29.57	29.34
D	15.73	15.54	15.49	15.42	15.39
T	11.71	11.64	11.50	11.51	11.48
C_B	0.681	0.729	0.730	0.730	0.730
V_k	14.00	12.25	12.32	12.21	12.48

Table 8: Computational results for minimizing lightship weight.

Objective	Parsons and Scott [24]	PSO	BPSO	NSPSO	NSBPSO
Transportation cost (T_c)	9.474	9.374	9.341	9.325	9.286
Lightship weight (L_s)	5240.3	5019.9	4946.6	4932.6	4809.2
Annual cargo (A_c)	386500	371960	369257	366855	361372
L	150.73	155.32	154.59	155.7	152.8
B	25.12	24.36	24.42	24.78	22.87
D	13.84	14.43	14.18	13.02	13.92
T	10.39	10.76	10.62	10.38	10.57
C_B	0.750	0.693	0.69	0.705	0.750
V_k	14.00	13.62	13.78	13.29	13.25

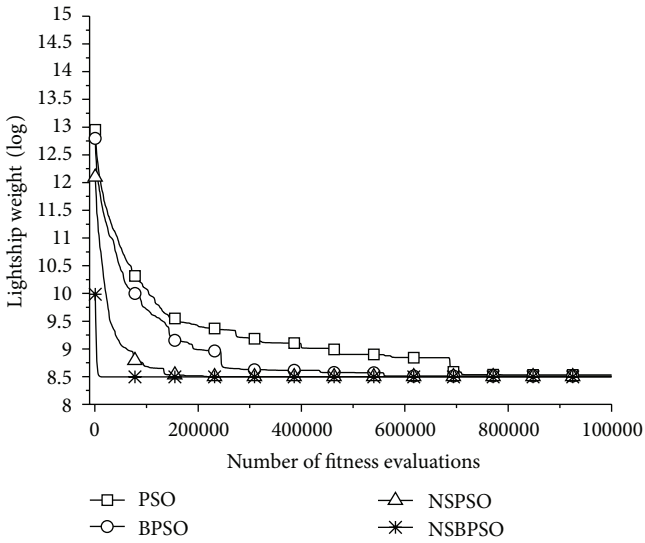

Figure 4: The convergence curves of PSO, BPSO, NSPSO, and NSBPSO for minimizing lightship weight.

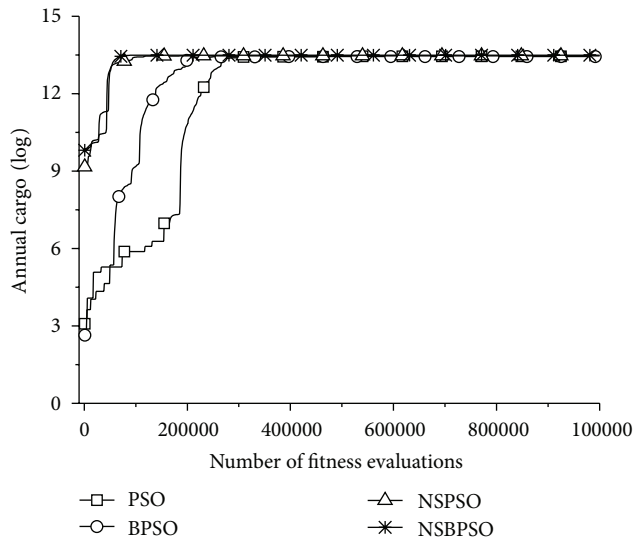

Figure 5: The convergence curves of PSO, BPSO, NSPSO, and NSBPSO for maximizing annual cargo.

almost parameter-free algorithm except for the probability of neighborhood search (P_{ns}).

Experimental studies are conducted on twelve well-known benchmark problems, including unimodal, multimodal, and rotated multimodal problems. Computational results show that the parameter P_{ns} does not affect the performance of NSBPSO. When $P_{\text{ns}} > 0$, NSBPSO can obtain good performance. Another comparison demonstrates that NSBPSO performs better than, or at least comparable to, several other state-of-the-art PSO algorithms. Compared to PSO with neighborhood search (NSPSO), our approach NSBPSO not only achieves better results, but also has less control parameters.

For the ship design problem, NSBPSO performs better than other three algorithms when optimizing a single objective. When considering all three objectives, we cannot

Table 9: Computational results for maximizing annual cargo.

Objective	Parsons and Scott [24]	PSO	BPSO	NSPSO	NSBPSO
Transportation cost (T_c)	10.294	10.884	11.308	11.59	12.213
Lightship weight (L_s)	12436	12558.2	12694.5	12757.4	12769.2
Annual cargo (A_c)	700533	706405	737952	741928	753137
L	222.49	235.86	231.95	235.27	238.2
B	32.31	32.20	32.32	32.14	32.31
D	15.73	16.60	17.29	16.62	15.73
T	11.71	11.42	11.70	11.70	11.71
C_B	0.750	0.720	0.750	0.750	0.750
V_k	18.00	18.09	18.73	18.68	18.89

determine which algorithm is the best. Because one algorithm only achieves better results than other algorithms on one or two objectives. To tackle this problem, we can use multiobjective optimization algorithms to optimize the three objectives at the same time. This will be investigated in the future work.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (no. 51209057).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[2] J. Mirapeix, P. B. García-Allende, O. M. Conde, J. M. Lopez-Higuera, and A. Cobo, "Welding diagnostics by means of particle swarm optimization and feature selection," *Journal of Sensors*, vol. 2012, Article ID 318038, 11 pages, 2012.

[3] Y. Hu, Y. Ding, and K. R. Hao, "An immune cooperative particle swarm optimization algorithm for fault-tolerant routing optimization in heterogeneous wireless sensor networks," *Mathematical Problems in Engineering*, vol. 2012, Article ID 743728, 19 pages, 2012.

[4] S. S. Patnaik and A. K. Panda, "Particle swarm optimization and bacterial foraging optimization techniques for optimal current harmonic mitigation by employing active power filter," *Applied Computational Intelligence and Soft Computing*, vol. 2012, Article ID 897127, 10 pages, 2012.

[5] C. Knievel and P. A. Hoeher, "On particle swarm optimization for MIMO channel estimation," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID 614384, 10 pages, 2012.

[6] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.

[7] K. E. Parsopoulos and M. N. Vrahatis, "UPSO-A unified particle swarm optimization scheme," in *Proceedings of International Conference on Computational Methods in Sciences and Engineering*, pp. 868–873, 2004.

[8] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[9] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[10] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[11] H. Wang, Z. Wu, S. Rahnamayan, C. Li, S. Zeng, and D. Jiang, "Particle swarm optimisation with simple and efficient neighbourhood search strategies," *International Journal of Innovative Computing and Applications*, vol. 3, no. 2, pp. 97–104, 2011.

[12] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[13] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 6, pp. 1362–1381, 2009.

[14] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol. 42, no. 3, pp. 627–646, 2012.

[15] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 80–87, 2003.

[16] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[17] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or cauchy jumps," in *Proceedings IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 3285–3291, May 2009.

[18] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 354–372, 2012.

[19] H. Wang, "Opposition-based barebones particle swarm for constrained nonlinear optimization problems," *Mathematical Problems in Engineering*, vol. 2012, Article ID 761708, 12 pages, 2012.

[20] F. Van Den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[21] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[22] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[23] P. Sen and J. B. Yang, *Multiple Criteria Decision Support in Engineering Design*, Springer, London, UK, 1988.

[24] M. G. Parsons and R. L. Scott, "Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods," *Journal of Ship Research*, vol. 48, no. 1, pp. 61–76, 2004.

[25] C. G. Hart and N. Vlahopoulos, "An integrated multidisciplinary particle swarm optimization approach to conceptual ship design," *Structural and Multidisciplinary Optimization*, vol. 41, no. 3, pp. 481–494, 2010.