IMPROVED PARTICLE SWARM OPTIMIZATION AND NEIGHBORHOOD FIELD OPTIMIZATION BY INTRODUCING THE RE-SAMPLING STEP OF PARTICLE FILTER

QIFENG CHENG*
Research and Development Center, China Academy of Launch Vehicle Technology
Fengtai District, Beijing 100076, China
Institute of Intelligent Engineering and Mathematics, Liaoning Technical University
Fuxin City, Liaoning Province 123000, China

XUE HAN AND TINGTING ZHAO
Institute of Intelligent Engineering and Mathematics, Liaoning Technical University
Fuxin City, Liaoning Province 123000, China

V. S. SARMA YADAVALLI
Department of Industrial and Systems Engineering, University of Pretoria
Pretoria 0002, South Africa

(Communicated by Chao Xu)

Abstract. A technique of introducing the re-sampling step of particle filter is proposed to improve the particle swarm optimization (PSO) algorithm, a typical global search algorithm. The re-sampling step can decrease particles with low weights and duplicate particles with high weights, given that we define a type of suitable weights for the particles. To prevent the identity of particles, the re-sampling step is followed by the existing method of particle variation. Through this technique, the local search capability is enhanced greatly in the later searching stage of PSO algorithm. More interesting, this technique can also be employed to improve another algorithm of which the philosophy is “learning from neighbors”, i.e., the neighborhood field optimization (NFO) algorithm. The improved algorithms (PSO-resample and NFO-resample) are compared with other metaheuristic algorithms through extensive simulations. The experiments show that the improved algorithms are superior in terms of convergence rate, search accuracy and robustness. Our results also suggest that the proposed technique can be general in the sense that it can probably improve other particle-based intelligent algorithms.

1. Introduction. Optimization is the process of finding the maximum/minimum solution for a given objective function with possibly a set of constraints. Since it plays a key role in practical problems, optimization is always an ongoing research topic. Certain local search methods, such as gradient-based search [14, 22] and tabu search algorithms [11], are successfully applied in conventional applications. But these local algorithms are easily getting trapped in local optima when facing much complicated problems. When the real-world problems become more complex,
some global search algorithms have been proposed to explore the whole search space stochastically, which are inspired by biological systems, such as genetic algorithm (GA) [12, 33], particle swarm optimization (PSO) [20, 10] and differential evolution (DE) [31, 24]. These algorithms can solve highly complicated problems and are more possible to escape from local optima and approach the global optima. The standard particle swarm optimization (PSO) is an intelligent optimization algorithm proposed by Kennedy and Eberhart [20]. The idea is derived from the study of birds’ simplified social model and the simulation of the migration and accumulation behavior in the process of the flock foraging. PSO employs the advantages of simple structure, easy to implement, fewer parameters to adjust. Hence it has been widely applied in both theoretical and applied research, such as combinatorial optimization, engineering design and optimization, model predictive control [40]. However, the standard PSO algorithm suffers from low precision and slow convergence speed, and can easily plunge into local optima. Therefore, various improved PSO algorithms have been studied and verified by the research community.

There are mainly three kinds of strategies to improve PSO algorithm. The first one is to adjust the algorithm parameters. In [30], a method involving a linearly decreasing inertia weight is presented, which aims at balancing the global and local search capabilities. A large inertia weight ensures that particles have larger velocities in the early searching stage, thereby strong global search ability is endowed. On the contrary, a strong local search in the later searching stage is enabled with a small inertia weight. [6] improves the convergence rate through adjusting the constriction factor in the searching process. The second strategy explores different types of topologies to elevate PSO’s performance. A dynamically adjusted neighborhood is applied in [32], where the neighborhood of a particle gradually increases until it includes all particles. A fully informed PSO is introduced by [27], where all the neighbors of the particle are used to update the velocity instead of using the pbest and gbest positions in the standard algorithm. In [29], the fitness-distance-ratio-based PSO (FDRPSO) with near neighbor interactions is developed. When updating each dimension of the velocity, a new item is accounted for, which implies each particle also absorbs the experience of the neighboring particles that have higher fitness values than itself. And a unified particle swarm optimizer (UPSO) is constructed in [28] by merging the global version and local version of PSO. The third strategy revolves around to interacting with other optimization methods or techniques. In [4], the collision-avoiding mechanisms are applied to relocate the particles, which effectively prevent particles from moving too close to each other in order to maintain the diversity and escape from local optima. In [1], a multi-objective evolutionary optimization algorithm using RBF-PSO has been applied. An hybridization of PSO and K-means algorithm with two simultaneous cost functions is proposed in [2], which employs a fuzzy approach. Novel improved algorithms are attracting ongoing attentions of research community [15, 39, 16, 17], which either belongs to one of the above three strategies or combines two of them. These improved PSO algorithms have circumvented the shortcomings of the standard algorithm, to different extent, and improved the performance.

In this paper we define a weight for the particle according to its distance from the current best particle, then the re-sampling step of particle filter can be employed in the later searching stage of PSO to duplicate particle with high weights and decrease particles with low weights. This can thus enhance the local search ability. Following the re-sampling step, the existing method of particle variation
is executed to avoid the identity of the particles. This re-sampling technique (plus the particle variation) lies in the third strategy described above. Moreover, it is very interesting to find that this technique can improve another newly developed searching algorithm, i.e., neighborhood field optimization (NFO) \cite{35, 36, 34}. The NFO has emulated the local phenomenon in the real world that each individual is mainly affected by its neighbors. And its philosophy is learning from neighbors \cite{35}. In NFO, each individual only shares information with its neighbors in the search space, and it will be attracted by its superior neighbor and repulsed by its inferior neighbor towards fitter regions. NFO utilizes exact neighbors to generate directions of search, which can approximate descending directions of the objective function. The theory of NFO is simple, and it is easy to implement. To further lift the performance of NFO, the re-sampling technique is employed in the later stage. We test the proposed algorithms under different benchmark functions. The simulation results verify the effectiveness and the significant performance improvement of the improved algorithms, compared with the standard PSO and NFO. Lastly, the emphasis is made to clarify the main contribution of the paper. It presents a feasible method to improve the standard PSO and NFO algorithms at the first glance. But more important, this paper proposes a general technique which can be exploited for many other particle-based intelligent algorithms. Our results and analyses imply that this technique can probably upgrade their performance.

The rest of the paper is proceeded as follows. Section 2 reviews the algorithms of the standard PSO and NFO. Section 3 first gives the introduction of particle filter and the re-sampling step, then discusses the strategy and implementation of the improved PSO and NFO. Section 4 presents the test functions and compares the simulation results. Finally, the paper is concluded in Section 5.

2. Related algorithms.

2.1. The standard PSO algorithm. The standard PSO algorithm is a population-based algorithm. Each particle in the search space is one solution of the problem and corresponds to one fitness value determined by the objective function. It flies at a certain speed (direction) during the iterative processes and the initial particles are a group of random ones. The path of each particle is evolved according to two “extremum”: one of which is the optimal solution among this particle’s trajectories and the other is the optimal solution of all particles’ trajectories. The mathematical formulation is as follows. In the search space, the population size of all particles is $N$, the $i$th particle’s position and speed are expressed as $x_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, $v_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ respectively. The optimal position among the $i$th particle’s trajectories is represented as $pBest_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, and the current global optimal location (the optimal among all particles’ trajectories) is denoted as $gBest = (p_{g1}, p_{g2}, \cdots, p_{gD})$. For the $k$th iteration, the position and velocity of the $i$th particle in the $j$th ($1 \leq j \leq D$) dimension is updated as

$$v_{ij}^{k+1} = w \times v_{ij}^k + c_1 x r_1 \times (p_{ij} - x_{ij}^k) + c_2 x r_2 \times (p_{gj} - x_{ij}^k),$$  \hspace{1cm} (1)$$

$$x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1}. \hspace{1cm} (2)$$

$w$ is the inertia weight factor, $c_1, c_2$ are acceleration constants, and $r_1, r_2$ are uniformly distributed random numbers between 0 and 1. $v_{ij}^k$ denotes the $j$th component of velocity $v_i$ at the $k$th iteration of the $i$th particle and the range of it is $[v_{min,j}, v_{max,j}]$. If the velocity computed by (1) exceeds this range, then the boundary value will be used.
2.2. Neighborhood field optimization algorithm. In NFO, each individual is updated under the philosophy of “learning from the neighbors” [35]. NFO solves the optimization problem by the cooperation and competition of particles. Each particle is influenced by its superior neighbors positively and by its inferior neighbors negatively, and then the mutation and crossover operations are used to reproduce another set of population to increase the diversity. Lastly, one-to-one selection based on greedy strategy is applied to the offspring and corresponding parent individuals, which leads to a new generation of population.

The randomly chosen $N$ particles are denoted by $X = (x_1, x_2, \cdots, x_N)$. At $G$th generation, each particle is encoded as $x_{i,G} = (x_{i1,G}, x_{i2,G}, \cdots, x_{iD,G})$, which is also called the target vector. The superior neighbor $x_{c_i,G}$ and the inferior neighbor $x_{w_i,G}$ of each particle can be found by

$$
\begin{align*}
xc_{i,G} &= \arg \min_{f(x_k,G)<f(x_i,G)} \|x_k,G - x_i,G\| \\
xw_{i,G} &= \arg \min_{f(x_k,G)>f(x_i,G)} \|x_k,G - x_i,G\|
\end{align*}
$$

where $\| \cdot \|$ is the distance evaluation, e.g., Euclidean distance. If $x_{i,G}$ is in the best individual in the population, $xc_{i,G}$ is defined as $x_{i,G}$; if $x_{i,G}$ is in the worst individual in the population, $xw_{i,G}$ is defined as $x_{i,G}$. Then, the mutation and crossover operations are conducted:

$$v_{i,G} = x_{i,G} + \alpha \times r_1 \times (xc_{i,G} - x_{i,G}) + \alpha \times r_2 \times (xc_{i,G} - xw_{i,G}),$$

$$u_{ij,G} = \begin{cases} v_{ij,G}, & \text{rand}(0,1) \leq Cr \text{ or } j = j_{rand} \\ x_{ij,G}, & \text{otherwise} \end{cases},$$

where $1 \leq i \leq N$, $1 \leq j \leq D$, $v_{i,G} = (v_{i1,G}, v_{i2,G}, \cdots, v_{iD,G})$ is the mutant vector and $u_{i,G} = (u_{i1,G}, u_{i2,G}, \cdots, u_{iD,G})$ is the trial vector of its target vector. $\alpha$ is the learning rate, $r_1$ and $r_2$ are two uniformly distributed random numbers in $[0,1]$, $Cr$ is the crossover probability, and $j_{rand}$ suggests that at least one random component will get value from the mutant vector so that the trial vector is different from the target vector. Comparing the trial vector with the corresponding target vector, the better one (with a lower fitness value) will be chosen into the next generation:

$$x_{i,G+1} = \begin{cases} u_{i,G}, & f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G}, & \text{otherwise} \end{cases},$$

where $f$ is the objective function.

3. The improved PSO and NFO algorithms.

3.1. The re-sampling step in particle filter. Particle filter is an approach to solve the state estimation problem, which uses a group of statistical samples to approximate the posterior probability density function given a series of measurements [3, 7]. It is an algorithm that implement recursive Bayesian estimation through non-parametric Monte Carlo methods [9]. Particle filter can be applied in general nonlinear, non-Gaussian cases in which the densities of some random variables are hard to evaluate. For example, to calculate the expectation over the sampling process of integrals of any function, which is tractable, can be used to estimate the integral with the exact density which is often intractable.

The most common way to generate samples in particle filter is importance sampling (IS). The importance sampling aims to sample the distribution according to
the “importance” of the region to advance computational efficiency. This is vital when the density distribution is sparse, i.e., the region which the target lies in is rather small across the support space. The IS employs a proposal distribution which is readily sampled to evaluate the original distribution which may be hard to sample [9]. While it is often difficult to choose a good proposal distribution, the sequential importance sampling (SIS) is then developed to mitigate this problem. The SIS update the proposal distribution sequentially online. However, both IS and SIS will face the problem of weight degeneracy, i.e., after several iterations, in additional to one/few sample, the importance weight of other samples are negligible (close to 0). Thus a lot of computation effort is wasted on the trivial weight samples, and the particles can no longer approximate the actual posterior probability distribution well. In order to circumvent the problem, the re-sampling is implemented [13]. The re-sampling method can not only duplicate particles with larger weights, lessen particles with smaller weights, but can ensure that the number of particles keeps the same.

In the later searching stage of PSO/NFO algorithm, if we weigh the particles according to their distances to the current global optimal solution (gBest), accordingly re-sampling step can be introduced to reproduce particles which are close to gBest and reduce particles which are far from gBest. This leads to a stronger local search ability, which is more emphasized in the later searching stage and thereby improves the efficiency of the algorithm. However, the re-sampling method will bring another problem, the so-called particle impoverishment, i.e., the diversity of particles is cut down. Thus, use is made of mutation operation after the re-sampling step to alleviate the sample impoverishment problem in the improved PSO/NFO algorithms.

There are a number of re-sampling methods available in the literature. Four common methods will be introduced here [18], i.e., simple random re-sampling, stratified re-sampling, systematic re-sampling and residual re-sampling. The basic principles of the re-sampling methods are similar. Assume there are a set of m particles \( \{z_i\} (i = 1, 2, \ldots, m) \) with weight \( \tau_i \), we need to select m particles \( \{z^*_j\} (j = 1, 2, \ldots, m) \) with uniform weights \( \tau^*_j = 1/m \) from the set \( \{z_i\} \) according to the condition

\[
P(z^*_j) = \begin{cases} 
\tau_i, & z^*_j = z_i \\
0, & z^*_j \neq z_i
\end{cases}
\]

If we view \( \{z^*_j\} \) to be the \( j \)th event which will copy some \( z_i \), then the subscript \( i \) will be determined by employing the inverse of the generalized cumulative density function of \( z_i \) to map a random number \( a_j \in [0, 1) \). Thus the number of \( z_i \) chosen to copy, \( n_i \), is equal to the number of \( a_j \in \{ \sum_{s=1}^{i-1} \tau_s, \sum_{s=1}^{i} \tau_s \} \). Different ways to generate the random number \( a_j \) construct different re-sampling methods.

**1) Simple random re-sampling (efficient) [8]**

Let \( \tau_i \) represent the normalized weight of the distance between particle \( z_i \) and the current optimal solution, gBest.

(1-i) Select \( m \) uniformly distributed random numbers \( \tilde{a}_j : \tilde{a}_j \sim U[0, 1) \), and then obtain a sequence of ordered uniform random numbers \( a_j \) according to

\[
a_j = \begin{cases} 
\frac{1/m}{a_m}, & j = m \\
\frac{1}{a_j a_{j-1}}, & j = 1, 2, \ldots, m - 1
\end{cases}
\]
(1-ii) Count \( n_i \), the number of copying \( z_i \), which is equal to the number of \( a_j \in (\sum_{s=1}^{i-1} \tau_s, \sum_{s=1}^{i} \tau_s) \).

(2) Stratified re-sampling \([5]\)

(2-i) Select \( m \) uniformly distributed random numbers \( \bar{a}_j : \bar{a}_j \sim U[0,1] \), and obtain a set of random numbers \( a_j \) that is in ascending order according to

\[
a_j = \frac{(j-1) + \bar{a}_j}{N}
\]  

(8)

(2-ii) Same as step (1-ii).

(3) Systematic re-sampling \([23]\)

(3-i) Select one uniformly distributed random number \( \bar{a} : \bar{a} \sim U[0,1] \), and obtain a set of random numbers \( a_j \) that is in ascending order according to

\[
a_j = \frac{(j-1) + \bar{a}}{N}
\]  

(9)

(3-ii) Same as step (1-ii).

(4) Residual re-sampling \([26]\)

Residual re-sampling differs from the above re-sampling methods, as it is a partially deterministic method. It proceeds as follows:

(4-i) For \( i = 1, 2, \ldots, m \), retain \( n'_i = \lfloor m \tau_i \rfloor \) copies of \( z_i \); thus the number of the remaining particles \( M = m - \sum_{i=1}^{m} n'_i \).

(4-ii) Using any of the above re-sampling methods (1)/(2)/(3), re-sample \( M \) particles from \( \{z_i\} \) with new weights proportional to \( m \tau_i - n'_i \).

It is stated that the computational complexity of the four re-sampling methods is comparable, i.e., they can all be implemented in time \( O(N) \) \([18]\). In the subsequent sections, we apply simple random re-sampling in the improved algorithms, but otherwise the procedure is similar using other re-sampling methods.

3.2. Improved algorithms. As illustrated above, in the later searching stage of PSO and NFO, the searching speed becomes slow. Therefore, the re-sampling step in particle filter is absorbed in the later searching stage so that the particles far from the current global optimum are reduced and those close to the optimum are retained and copied. In this way, the search space is narrowed to improve the convergence rate and accuracy of solution. Yet the re-sampling may cause the loss of particle diversity, mutation operation (particle variation) is conducted after the re-sampling to mitigate the problem. In this section, we will present the detailed algorithms, which are named PSO-resample and NFO-resample respectively. The flowcharts are shown in Figure 1 and 2.

3.2.1. The improved PSO algorithm (PSO-resample). Step 1. Initialization

(a) Set constant parameters: the scale of particle swarm \( N \), the inertia weight \( w \), the acceleration constants \( c_1, c_2 \), the boundary of particle velocities \([v_{\text{min}}, v_{\text{max}}]\). Choose the re-sampling frequency \( \delta \), i.e., the re-sampling is executed every \( \delta \) step.

(b) Set the initial iteration step \( k = 0 \); randomly initialize particles’ positions \( x_i^k \) and particle velocities \( v_i^k \), where \( i \) represents the \( i \)th particle in the swarm and \( i = 1, \ldots, N \). Evaluate the fitness values of the initial particles in order to obtain the optimal position of each particle and the swarm, \( p\text{Best}_i^k \) and \( g\text{Best}^k \).
Step 2. If $k \geq k_1$ ($k_1$ is the scheduled iteration step), when the algorithm enters the later searching stage, go to Step 3; otherwise go to Step 6.

Step 3. If $k-k_1\delta$ is an integer, implement Step 4; otherwise go to Step 6.

Step 4. Compute the normalized weight of each particle $x_i^k$ according to its distance from the current best particle of the swarm, $gBest^k$:

$$F_i = \exp(- \| x_i^k - gBest^k \|_2), \quad \tau_i = \frac{F_i}{\sum_{i=1}^{N} F_i}, \quad (10)$$
for \( i = 1, 2, \ldots, N \). Then implement the re-sampling by the simple random re-sampling procedure (1-i) and (1-ii).

**Step 5.** If the particles are obtained by copying in the re-sampling step, i.e., there are at least 2 identical particles, then implement the particle variation \[37\]: get \( \eta \) from the Gaussian distribution \((0, 1)\), update the particles by \( x_i^k = x_i^k \times (1 + 0.5 \times \eta) \). Calculate the fitness value and obtain the optimal position of each particle and the swarm, \( pBest_k^i \) and \( gBest_k \).

**Step 6.** Get particles’ velocities \( v_{i,G}^{k+1} \) and positions \( x_{i,G}^{k+1} \) based on (1) and (2), compute the fitness value and obtain \( pBest_{i,G}^{k+1} \) and \( gBest_{G+1} \).

**Step 7.** If the termination condition is satisfied, then the algorithm stops and outputs the best swarm position \( gBest_{G+1} \); otherwise \( G = G + 1 \), go to Step 2.

3.2.2. The improved NFO algorithm (NFO-resample). **Step 1.** Initialization

(a) Set constant parameters: the population size \( N \), the learning rate \( \alpha \), the crossover probability \( C_r \). Choose the re-sampling frequency \( \delta \).

(b) Set the initial iteration step \( G = 0 \); randomly initialize particles’ positions \( x_{i,G}^i \), where \( i \) represents the \( i \)th particle and \( i = 1, \ldots, N \).

**Step 2.** If \( G \geq G_1 \) \( (G_1 \) is the scheduled iteration step), when the algorithm enters the later searching stage, go to Step 3; otherwise go to Step 6.

**Step 3.** If \( \frac{G - G_1}{\delta} \) is an integer, implement Step 4; otherwise go to Step 6.

**Step 4.** Calculate the fitness value and obtain the optimal position of the swarm at the \( G \)th generation, \( gBest_G \). Compute the normalized weight of each particle \( x_{i,G} \) according to its distance from the current best particle, \( gBest_G \):

\[
F_i = \exp(- \| x_{i,G} - gBest_G \|_2), \quad \tau_i = \frac{F_i}{\sum_{i=1}^{N} F_i},
\]

for \( i = 1, 2, \ldots, N \). Then implement the re-sampling by the simple random re-sampling procedure (1-i) and (1-ii).

**Step 5.** If the particles are obtained by copying in the re-sampling step, i.e., there are at least 2 identical particles, then implement the particle variation \[37\]: get \( \eta \) from the Gaussian distribution \((0, 1)\), update the particles by \( x_{i,G} = x_{i,G} \times (1 + 0.5 \times \eta) \).

**Step 6.** Update the particles’ positions \( x_{i,G+1} \) by localization, mutation, crossover and selection, i.e., by Equation (3), (4), (5) and (6).

**Step 7.** If the termination condition is satisfied, then the algorithm stops and outputs the best swarm position \( gBest_{G+1} \); otherwise \( G = G + 1 \), go to Step 2.

It is noted that the construction of the improved NFO algorithm is similar to that of the improved PSO algorithm, where the difference is the particle update mechanism. This illustrates that the technique of introducing the re-sampling step of particle filter can be embedded into other particle-based algorithms and tested in a similar way, which can be easily implemented.

4. Simulation results and analysis. In this section, we compare the performance of the standard PSO with PSO-resample, the standard NFO with NFO-resample and seven other metaheuristics through minimizing the benchmark functions. As it is shown in Table 2, twenty test functions that are always termed as
Table 1. Parameter settings

| Algorithm            | Parameters                                      |
|----------------------|-------------------------------------------------|
| Improved PSO         | $c_1 = c_2 = 2, w = 0.7298$                     |
| Standard PSO         | $c_1 = c_2 = 2, w = 0.7298$                     |
| Improved NFO         | $\alpha = 1.3, Cr = 0.1$                        |
| Standard NFO         | $\alpha = 1.3, Cr = 0.1$                        |
| PSO-cf-local         | $c_1 = c_2 = 2, w = 0.7298$                     |
| UPSO                 | $c_1 = c_2 = 2, w = 0.7298, u = 0.1$            |
| FDRPSO               | $c_1 = c_2 = 1, c_3 = 2$                        |
| CLPSO                | $c_1 = c_2 = 2, w : 0.9 \sim 0.4, m = 0.7$     |
| LDWPSO               | $c_1 = c_2 = 2, w : 0.9 \sim 0.4$              |
| DE                   | $F = 0.5, CR = 0.9$                             |

classical benchmarks are used to verify the performance. They include 7 uni-modal functions ($f_1 \sim f_7$) and 13 multi-modal functions ($f_8 \sim f_{20}$), and a more detailed description of them can be found in [38]. The metaheuristics against which the proposed algorithms are evaluated are: FDRPSO [29], UPSO [28], CLPSO (comprehensive learning PSO) [25], PSO-cf-local (local version of PSO with constriction factor) [21], LDWPSO (linearly decreasing inertia weight PSO) [30], DE [31, 24]. And the parameters for the algorithms are chosen as in Table 1. For both FDRPSO and LDWPSO, $w$ shows a linear decrease from 0.9 to 0.4.

The group size $N$ is chosen to be 30. For each algorithm, we try two different dimensions for test functions, 20 and 40. The range of the velocity for PSO and all its variants, $[v_{\text{min}, j}, v_{\text{max}, j}]$, is set to be 20% of the range $x_i$, as shown in Table 2. Each algorithm for each test function is implemented 25 times independently, and the average fitness values and variances are calculated to evaluate the performance of the algorithms. Specifically, the average fitness value reflects the precision of the result and the variance reflects the stability and robustness of the algorithm.

There are two adjustable parameters in the improved algorithms, i.e., the re-sampling frequency $\delta$ and the scheduled iteration step $k_1/G_1$. In order to test the sensitivity of the parameters, extensive experiments are carried out. A same termination condition is employed across the experiments of the paper that the maximum iterations reaches 2000. Table 3 shows the means and variances under PSO-resample with different $\delta$ and $k_1$ for the $D=20, N=30$ case, where the benchmark functions $f_1, f_5, f_8, f_{10}, f_{13}, f_{15}, f_{18}, f_{19}$ are tested. The results of other functions are omitted here due to space limitation.

For function $f_{10}$ and $f_{13}$, the best means and smallest variances both appear when $\delta=20$ and $k_1=100$. For function $f_1, f_{15}, f_{18}$, the results when applying different parameters are nearly the same since the errors are negligible. For function $f_5, f_8$, the best means are achieved when $\delta=20$ and $k_1=100$, and the variances are 0.2943 and 21.2570 respectively, which are 35.8% and 13.5% larger than the smallest one. For $f_{19}$, the mean when $\delta=20, k_1=100$ is 3.6% worse than the best result, and the variance is 18.5% larger. Therefore, on the whole, the optimal parameters among these trials are $\delta=20$ and $k_1=100$. $\delta=20$ is reasonable by taking into account that re-sampling frequently (small $\delta$) will result in a premature convergence and re-sampling infrequently (big $\delta$) will weaken the effect of the re-sampling step.
Table 2. Benchmark functions

| $f_i$                                                                 | Range of $x_i$ | Theoretical optimum |
|----------------------------------------------------------------------|----------------|---------------------|
| $f_1 = \sum_{i=1}^{D} x_i^2$                                         | $[-100,100]$   | 0                   |
| $f_2 = \sum_{i=1}^{D} x_i + \prod_{i=1}^{D} |x_i|$                   | $[-10,10]$      | 0                   |
| $f_3 = \sum_{i=1}^{D} (\sum_{j=1}^{D} x_j)^2$                         | $[-100,100]$   | 0                   |
| $f_4 = \max_{1 \leq i \leq D} \{x_i\}$                               | $[-100,100]$   | 0                   |
| $f_5 = \sum_{i=1}^{D} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2)$           | $[-100,100]$   | 0                   |
| $f_6 = \sum_{i=1}^{D} (x_i + 0.5)^2$                                  | $[-100,100]$   | 0                   |
| $f_7 = \sum_{i=1}^{D} x_i^4 + \text{random}(0,1)$                    | $[-1.28,1.28]$ | 0                   |
| $f_8 = \sum_{i=1}^{D} x_i^2 - 10 \cos(2\pi x_i) + 10$                | $[-5.12,5.12]$ | 0                   |
| $f_9 = 20 + e - 20\exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i))$ | $[-30,30]$     | 0                   |
| $f_{10} = \sum_{i=1}^{D} x_i^2 - \sum_{i=1}^{D} \cos(x_i)$ + 1        | $[-600,600]$   | 0                   |
| $f_{11} = \frac{1}{\sqrt{D}} \left[10 \sin^2(x_{i1}) + \sum_{i=1}^{D} (y_i - 1)^2 \cdot [1 + 10\sin^2(x_{i+1})]\right]$ | $[-50,50]$     | 0                   |
| $f_{12} = 0.1 \left[10 \sin^2(x_{i1}) + \sum_{i=1}^{D} (y_i - 1)^2 \cdot [1 + 10\sin^2(x_{i+1})]\right]$ | $[-50,50]$     | 0                   |
| $f_{13} = \left(\frac{1}{100} + \sum_{j=1}^{D} x_i \right)^{-1}$     | $[-65.536, 65.536]$ | 1                   |
| $f_{14} = \frac{11}{\pi} \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) $ | $[-5.5]$       | 0.0003075           |
| $f_{15} = x_1^2 - 2x_1^4 + \frac{1}{4} x_1^6 + x_1 x_2 - 4x_2$      | $[-5.5]$       | -1.03163            |
| $f_{16} = (x_2 - \frac{1}{\sqrt{16}} x_1^2 + \frac{1}{2} x_1 - 6)^2 + (1 - \frac{1}{\sqrt{\pi}}) \cos x_1 + 10$ | $[-5.10] \times [0.15]$ | 0.398 |
| $f_{17} = (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 + 2x_2^2)) \times [10 + (2(1 - x_1)^2)x_1 + 10(x_2 - 3x_1^2 + 2x_2^2)]$ | $[-2, 2]$      | 3                   |
| $f_{18} = -\sum_{j=1}^{D} \exp[-\frac{1}{2} x_j (x_j - 1)]$            | $[0, 1]$       | -3.32               |
| $f_{19} = -\sum_{j=1}^{D} (x_j - 1)'^2$                               | $[0, 10]$      | -10                 |
| $f_{20} = -\sum_{j=1}^{D} (x_j - 1)^2$                                | $[0, 10]$      | -10                 |

Table 3: Means and variances under PSO-resample, with different $\delta$ and $k_1$

| $f$ | $k_1$ | $\delta = 10$ | $\delta = 20$ | $\delta = 30$ | $\delta = 40$ |
|-----|-----|-------------|-------------|-------------|-------------|
| $f_1$ | 80  | 1.1697e-23 | 1.4034e-22 | 1.3997e-19 | 1.5248e-17 |
|      |     | variance   | 3.1624e-45 | 1.7331e-43 | 1.9198e-37 | 9.7892e-34 |
| 100  | mean | 1.2345e-24 | 3.4313e-23 | 6.7734e-20 | 1.3077e-17 |
|      | variance | 1.2889e-47 | 4.0997e-45 | 5.7511e-38 | 1.7539e-33 |
| 150  | mean | 4.8036e-24 | 5.5423e-22 | 4.7328e-19 | 1.5124e-17 |
|      | variance | 5.3061e-46 | 1.6406e-42 | 1.4786e-36 | 4.8891e-34 |
| 200  | mean | 3.4374e-23 | 2.2118e-21 | 8.3626e-19 | 9.8416e-17 |
|      | variance | 2.0994e-44 | 1.6926e-41 | 3.6846e-36 | 5.8052e-32 |
| 300  | mean | 3.7652e-23 | 8.6217e-21 | 9.6451e-18 | 1.7330e-16 |
### Table 1: Improved PSO & NFO by Introducing the Re-Sampling Step

|          | $f_5$ mean variance |         |         |         |
|----------|---------------------|---------|---------|---------|
|          | 1.6413e-44          | 3.3834e-40 | 6.4054e-34 | 8.3178e-32 |
|         | 23.0984             | 15.5308 |
|          | 0.1502e+03          | 0.2934  |
|          | 0.8921              | 132.7182 |
|          | 15.4714             | 15.5302 |
|          | 15.5411             | 15.5168 |
|          | 0.4272              | 0.9433  |
|          | 0.7471              | 1.0840  |
|          | 15.7879             | 15.9301 |
|          | 15.8046             | 1.6317  |
|          | 1.9228              | 132.4126 |
|          | 15.9560             | 17.9298 |
|          | 1.3168              | 0.5490  |
|          | 119.4753            | 236.7976 |

|          | $f_8$ mean variance |         |         |         |
|----------|---------------------|---------|---------|---------|
|          | 14.2808             | 13.4628 |
|          | 1.3502e+03          | 0.2934  |
|          | 0.3979              | 0.4585  |
|          | 0.8921              | 132.7182 |
|          | 14.1719             | 12.9326 |
|          | 14.8875             | 13.0902 |
|          | 20.5411             | 37.4915 |
|          | 15.5786             | 13.6945 |
|          | 37.2106             | 59.1946 |
|          | 16.3959             | 15.1895 |
|          | 35.1016             | 42.1339 |
|          | 0.0093              | 0.0089  |
|          | 0.0002              | 0.0064  |
|          | 5.5534e-04          | 3.3844e-04 |
|          | 4.5978e-04          | 2.2374e-04 |
|          | 0.0165              | 0.0157  |
|          | 0.0018              | 5.5528e-04 |
|          | 0.0328              | 0.0179  |
|          | 0.0098              | 5.0709e-04 |
|          | 0.1185              | 0.0489  |
|          | 0.0615              | 0.0124  |
|          | 0.0028              | 0.0079  |
|          | 5.0709e-04          | 0.0010  |
|          | 0.1885              | 0.0489  |
|          | 0.0615              | 0.0124  |
|          | 0.0098              | 5.0709e-04 |
|          | 0.1185              | 0.0489  |
|          | 0.0615              | 0.0124  |
|          | 0.0028              | 0.0079  |
|          | 5.0709e-04          | 0.0010  |
|          | 0.1885              | 0.0489  |
|          | 0.0615              | 0.0124  |
|          | 0.0028              | 0.0079  |
|          | 5.0709e-04          | 0.0010  |

|          | $f_{10}$ mean variance |         |         |         |
|----------|------------------------|---------|---------|---------|
|          | 1.1567                 | 1.2361 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |
|          | -1.1567                | 1.4680 |
|          | 0.2897                 | 0.4157 |
|          | 1.0774                 | 0.1512 |
|          | 0.5291                 | 0.1512 |

|          | $f_{15}$ mean variance |         |         |         |
|----------|------------------------|---------|---------|---------|
|          | 5.3248e-32             | 9.6635e-32 |
|          | 3.6682e-31             | 7.8886e-32 |
|          | -1.0316                | -1.0316 |
|          | 3.6682e-31             | 7.8886e-32 |
|          | -1.0316                | -1.0316 |
|          | 5.5220e-32             | 7.2970e-32 |
|          | -1.0316                | -1.0316 |
|          | 5.5220e-32             | 7.2970e-32 |
|          | -1.0316                | -1.0316 |

**Note:** The table lists the mean and variance values for different functions ($f_5$, $f_8$, $f_{10}$, $f_{13}$, $f_{15}$) and test cases (80, 100, 150, 200, 300) with the specified features (mean, variance).
Table 4. Convergence steps under PSO-resample

|   | f18 |   | f19 |   |   |   |   |   |   |   |
|---|-----|---|-----|---|---|---|---|---|---|---|
| 300 | mean | -1.0316 | -1.0316 | -1.0316 | -1.0316 |
|     | variance | 9.0719e-32 | 1.3213e-31 | 1.4397e-31 | 1.2622e-31 |
| 200 | mean | -3.8628 | -3.8628 | -3.8628 | -3.8628 |
|     | variance | 6.4394e-21 | 9.9421e-28 | 2.7894e-30 | 3.6522e-30 |
| 100 | mean | -3.8628 | -3.8628 | -3.8628 | -3.8628 |
|     | variance | 2.0992e-24 | 4.3654e-28 | 2.8872e-30 | 3.2895e-30 |
| 150 | mean | -3.8628 | -3.8628 | -3.8628 | -3.8628 |
|     | variance | 8.4359e-28 | 8.4809e-28 | 3.0844e-30 | 3.5420e-30 |
| 200 | mean | -3.8628 | -3.8628 | -3.8628 | -3.8628 |
|     | variance | 3.2422e-30 | 3.4394e-30 | 3.5814e-30 | 4.2914e-30 |
| 300 | mean | -3.8628 | -3.8628 | -3.8628 | -3.8628 |
|     | variance | 3.2422e-30 | 3.4394e-30 | 3.5814e-30 | 4.2914e-30 |

For all functions, we use a same criteria for judging convergence for all algorithms. If the difference between the best objective value $g_{Best}^k$ of the current generation $k$ and the average of the best objective value $g_{Best}^{k-1}, \ldots, g_{Best}^{k-k_{last}}$ of the last $k_{last}$ generation is equal to or less than $10^{-8}$, that is $|g_{Best}^k - \frac{1}{k_{last}} \sum_{i=k-k_{last}}^{k-1} g_{Best}^i| \leq 10^{-4}$ [19], we regard that the algorithm converges.

Table 4 shows the convergence rate under PSO-resample with different re-sampling frequency $\delta$ and a same scheduled iteration step $k_1 = 100$. It can be seen that, for a same $k_1$, the smaller the re-sampling frequency, the earlier the algorithm will converge except function $f_4, f_7, f_{13}, f_{15}, f_{17}, f_{18}, f_{19}, f_{20}$. In fact, for $f_4, f_{14} \sim f_{17}$, the convergence step is very close for different $\delta$. This is in accordance with our
Figure 3. Comparison of the convergence rate using different test functions ($D = 20, N = 30$)
judgement that re-sampling frequently will lead to a fast convergence, although it may not converge to the true optima.

Figure 3 reflects the convergence properties of different test functions with $D=20$, $N=30$ under four algorithms. Under the convergence criteria above, the situation for the standard and improved NFO is relatively simpler. The standard NFO converges at $k=337, 376, 428, 47, 120, 204, 174, 138, 435, 440, 321, 318, 40, 51, 16, 61, 29, 86, 31, 42$. While NFO-resample converges at step $k=252, 237, 54, 35, 509, 182, 53, 86, 31, 42$. In these set of functions except $f_5, f_{13}, f_{15}, f_{17}, f_{19}, f_{20}$, the convergence speed of NFO-resample is much faster than the standard NFO. We can see from Table 5 that the precision of solution of NFO-resample is better than that of the standard NFO. The comparison between the standard and improved PSO is similar to the NFO and NFO-resample. The comparison between the standard and improved PSO is more complex. For function $f_1, f_3$, the standard PSO does not converge and PSO-resample converges at $k=432, 794$. And for function $f_{11}, f_{12}$, the standard PSO converge at $k=1617, 1333$ and PSO-resample does not converges. And on function $f_5, f_8$, neither the standard PSO nor the improved PSO converges. The standard PSO converges at step $k=895, 74, 445, 255, 863, 1024, 117, 146, 46, 67, 54, 76, 96, 90$ for all functions except $f_1, f_3, f_5, f_8, f_{11}, f_{12}$ respectively, and for function $f_2, f_4, f_6, f_7, f_9$, $f_{10}, f_{13} \sim f_{20}$, PSO-resample converges at $k=537, 75, 348, 87, 648, 746, 160, 149, 44, 46, 55, 97, 107, 93$, which is earlier than the standard PSO except for $f_4, f_{13}, f_{14}, f_{17} \sim f_{20}$. Hence, as a whole, the PSO-resample and NFO-resample converges faster than the standard PSO and NFO respectively.

Table 5 presents the statistical results of 25 runs with the case of $D=20$, $N=30$ under four algorithms. It is observed that the average fitness values under PSO-resample, NFO-resample are better than those under the standard PSO, NFO respectively for most functions, and the variances are smaller. For function $f_{15}, f_{16}, f_{17}$, the results under four algorithms are nearly the same. For all functions except $f_5, f_8, f_{11}$, the average fitness values of PSO-resample and NFO-resample can approach their theoretical optimal values. However, it is shown that the means under PSO-resample and NFO-resample are much closer to the optima than those under the standard algorithms for $f_5, f_8, f_{11}$. Thus it can be concluded, no matter for uni-modal or multi-modal function, the accuracy of the solution, the global search ability and the stability of the results of PSO-resample and NFO-resample are all significantly better than the standard PSO and NFO respectively.

Table 6 shows the average computational time for a same maximum iteration step 2000, using Matlab 2013b on a CPU Intel Core i5, 1.70 GHz with a 8GB of RAM. It is noted that PSO-resample takes about two times longer than the standard PSO for a same maximum step (2000), while the computational cost of the improved and standard NFO is very close. The reason is that the re-sampling step weighs higher in the PSO algorithms than in the NFO, since the standard NFO takes much longer than the standard PSO. We also compute the number of function evaluations (NFE) for the PSO algorithms. For the improved PSO, the NFE is 126013, 126040, 125802, 126971, 125607, 125762, 125993, 113507, 125774, 125783, 125760, 125765, 125759, 125759, 125759, 125759, 125759, 125759, 125759, 125759, 125759, 125759, 125759. For the standard PSO, the number is 62247, 61048, 62272, 60182, 62255, 60131, 110430, 62178, 62148, 62219, 62199, 62087, 60078, 61066, 60079, 62031, 60084, 60258, 60154, 60156. The more the number of function evaluations is, the longer the computational time is.
Table 5. Means and variances on $D = 20$ test functions for $N = 30$ (25 runs)

| Function | PSO-resample | Standard PSO |
|----------|--------------|--------------|
| $f_1$    | 3.4313e-23(4.0997e-45) | 7.5386(24.3743) |
| $f_2$    | 3.1292e-13(4.2561e-25) | 2.5043(4.7289) |
| $f_3$    | 4.8851e-11(8.4971e-21) | 49.8877(710.3746) |
| $f_4$    | 1.5284e-94(5.2777e-187) | 1.5390e-54(2.3745e-107) |
| $f_5$    | 15.4349(0.4585) | 4.4216e+03(5.177e+07) |
| $f_6$    | 0(0) | 68.8400(2.1942e+03) |
| $f_7$    | 0.5155(0.0432) | 0.6040(0.0987) |
| $f_8$    | 11.2533(21.2570) | 27.3944(104.6841) |
| $f_9$    | 1.0509e-11(2.3499e-22) | 4.3043(2.0710) |
| $f_{10}$ | 0.0064(3.7421e-04) | 0.8194(0.0562) |
| $f_{11}$ | 1.6987(5.8474) | 5.5496(11.4830) |
| $f_{12}$ | -0.2002(0.1572) | 1.3155(0.5282) |
| $f_{13}$ | 1.0774(0.1512) | 1.0774(0.1512) |
| $f_{14}$ | 0.0035(3.6803e-05) | 0.0035(3.6803e-05) |
| $f_{15}$ | -1.0316(1.0255e-31) | -1.0316(1.9722e-31) |
| $f_{16}$ | 0.3979(0) | 0.3979(0) |
| $f_{17}$ | 3.0000(8.1600e-29) | 3.0000(2.0905e-30) |
| $f_{18}$ | -3.2789(0.0032) | -3.2694(0.0035) |
| $f_{19}$ | -8.4421(7.9474) | -6.8474(12.3691) |
| $f_{20}$ | -9.8865(3.1789) | -8.2623(11.7828) |

| Function | NFO-resample | Standard NFO |
|----------|--------------|--------------|
| $f_1$    | 5.0244e-38(6.0588e-74) | 1.0370e-33(2.5810e-65) |
| $f_2$    | 2.2622e-23(1.2282e-44) | 1.1769e-20(3.3240e-39) |
| $f_3$    | 1.3053e-07(4.0837e-13) | 5.7124(747.2053) |
| $f_4$    | 6.3508e-154(9.6797e-306) | 2.8710e-35(1.9782e-68) |
| $f_5$    | 2.3160(12.4777) | 16.3890(2.6798e+03) |
| $f_6$    | 0(0) | 0(0) |
| $f_7$    | 0(0) | 0(0) |
| $f_8$    | 0.1368(0.0049) | 0.2147(0.0059) |
| $f_9$    | -6.0396e-16(1.9387e-30) | 1.3465e-14(4.8468e-30) |
| $f_{10}$ | 0(0) | 0(0) |
| $f_{11}$ | 0.8247(1.1210e-15) | 0.8247(1.9722e-31) |
| $f_{12}$ | -1.1504(3.6713e-12) | -1.1504(1.9722e-31) |
| $f_{13}$ | 0.9980(3.5922e-22) | 0.9980(0) |
| $f_{14}$ | 9.5850e-04(9.2612e-11) | 0.0044(1.0963e-10) |
| $f_{15}$ | -1.0316(2.0116e-31) | -1.0316(1.9722e-31) |
| $f_{16}$ | 0.3979(1.7588e-14) | 0.3979(0) |
| $f_{17}$ | 3.0000(1.5108e-09) | 3.0000(3.1554e-30) |
| $f_{18}$ | -3.3215(1.0546e-10) | -3.3215(5.9953e-31) |
| $f_{19}$ | -10.1532(6.5141e-17) | -10.1532(2.9579e-09) |
| $f_{20}$ | -10.4029(2.9297e-12) | -10.4029(1.0758e-11) |

Table 7 gives the average fitness values and variances over 25 runs for $D=40$, $N=30$. Compared with Table 5, it is clear that a smaller dimension with the same population size will lead to a more accurate solution and a more stable result for the same algorithm, almost for all the functions, though there are several exceptions.
Table 6. Average computational time (seconds) for each run on $D = 20$ test functions for $N = 30$

| Function | PSO-resample | Standard PSO | NFO-resample | Standard NFO |
|----------|--------------|--------------|--------------|--------------|
| $f_1$    | 1.20         | 0.49         | 23.65        | 25.31        |
| $f_2$    | 1.30         | 0.52         | 25.61        | 25.26        |
| $f_3$    | 4.35         | 1.77         | 666.50       | 643.08       |
| $f_4$    | 2.94         | 0.91         | 50.79        | 54.41        |
| $f_5$    | 1.71         | 0.62         | 31.57        | 31.53        |
| $f_6$    | 1.49         | 0.51         | 26.43        | 24.78        |
| $f_7$    | 4.02         | 1.30         | 76.40        | 73.21        |
| $f_8$    | 1.85         | 0.74         | 33.34        | 35.71        |
| $f_9$    | 2.46         | 1.05         | 60.64        | 50.17        |
| $f_{10}$ | 2.57         | 1.01         | 52.85        | 57.37        |
| $f_{11}$ | 6.12         | 2.58         | 158.61       | 149.46       |
| $f_{12}$ | 4.42         | 2.22         | 129.69       | 127.31       |
| $f_{13}$ | 5.32         | 2.49         | 184.21       | 172.35       |
| $f_{14}$ | 1.71         | 0.72         | 42.70        | 42.96        |
| $f_{15}$ | 1.27         | 0.54         | 56.11        | 28.77        |
| $f_{16}$ | 1.06         | 0.47         | 22.90        | 20.07        |
| $f_{17}$ | 1.29         | 0.49         | 28.37        | 25.47        |
| $f_{18}$ | 2.09         | 0.84         | 47.97        | 50.29        |
| $f_{19}$ | 21.98        | 9.84         | 549.53       | 551.62       |
| $f_{20}$ | 2.59         | 1.16         | 133.86       | 120.27       |

which are displayed subsequently. For $f_{10}$, PSO-resample generates less accurate results for a smaller dimension. On function $f_{6}$, $f_{13}$, $f_{15}$, $f_{16}$, PSO-resample delivers a same result for the 20-D and 40-D problem. For $f_{6}$, $f_{12}$, $f_{13}$, $f_{15}$ $f_{20}$, the standard NFO and NFO-resample produce identical results for the 20-D and 40-D problem. For the standard NFO algorithm on $f_{14}$, the results are equal when $D = 20$ and $D = 40$. And it is noted for $f_{11}$, both the standard NFO and improved NFO produce more accurate results for a bigger dimension.

Table 8 presents the means and variances of the 25 runs under PSO-resample, NFO-resample and six other metaheuristic algorithms on the test functions with $D=20$. These eight algorithms achieve the same results on function $f_{13}$, $f_{16}$, $f_{17}$. NFO-resample performs best on function $f_{1}$, $f_{2}$, $f_{5}$ $f_{10}$ followed by other algorithms. PSO-resample has the best average fitness value on function $f_{3}$, $f_{6}$, $f_{12}$, $f_{20}$. For $f_{11}$, NFO-resample, UPSO, FDRPSO, CLPSO have the same better results. On function $f_{13}$, NFO-resample, UPSO, FDRPSO, CLPSO and DE achieve better results. For function $f_{18}$, the average fitness values under NFO-resample, UPSO, CLPSO all approach the theoretical optimum value. And UPSO achieves the best results on function $f_{14}$, $f_{19}$.

Specially, we compare PSO-resample, NFO-resample. It can be found that for function $f_{1}$, $f_{2}$, $f_{3}$, $f_{4}$, $f_{5}$, $f_{15}$, $f_{16}$, $f_{17}$, the two algorithms can all approach the optimal value. Therefore, it can be concluded here, on the whole, PSO-resample and NFO-resample are superior algorithms in terms of the accuracy of the solution, the global search ability and the stability of the results. As discussed above, the execution of PSO-resample takes longer than the standard PSO, nevertheless it is fairly acceptable from Table 6. And the execution of NFO-resample takes as long as the standard NFO.
Table 7. Means and variances on $D = 40$ test functions for $N = 30$ (25 runs)

| Function | PSO-resample | Standard PSO | NFO-resample | Standard NFO |
|----------|--------------|--------------|--------------|--------------|
| $f_1$    | 1.5796e-13(5.5281e-26) | 185.5670(5.1824e+03) | 3.1264e-20(2.3457e-38) | 1.2589e-14(3.8034e-27) |
| $f_2$    | 6.0313e-09(3.9308e-17) | 15.9041(29.1664) | 1.5768e-13(5.9668e-25) | 4.2327e-10(4.2998e-18) |
| $f_3$    | 5.7985e-04(4.9215e-07) | 1.7963e+03(4.6628e+05) | 1.0969e-04(1.9506e-07) | 890.1068(7.2077e+06) |
| $f_4$    | 3.6273e-92(3.1490e-182) | 1.9099e-53(3.6487e-105) | 3.0444e-161(2.2248e-320) | 7.1526e-33(1.2278e-63) |
| $f_5$    | 36.4677(0.5677) | 8.1082e+05(3.6749e+11) | 22.7611(25.6539) | 27.7804(146.3549) |
| $f_6$    | 0(0) | 950.9600(7.5934e+04) | 0.3979(0) | 0.3979(0) |
| $f_7$    | 0.6291(0.0868) | 0.6804(0.1075) | 3.0000(9.1551e-28) | 3.0000(2.0984e-30) |
| $f_8$    | 53.1581(129.8373) | 85.4913(724.0781) | 0.9274(0.1512) | 1.4333(1.2554) |
| $f_9$    | 0.0042(4.1661e-04) | 7.3142(2.9770) | 0.0038(3.3734e-05) | 0.0052(6.1041e-05) |
| $f_{10}$ | 7.7593e-04(6.0020e-06) | 2.1590(0.6044) | -1.0316(1.0255e-31) | -1.0316(1.9722e-31) |
| $f_{11}$ | 5.7450(10.1131) | 16.2519(46.5782) | 3.0000(9.1551e-28) | 3.0000(2.0984e-30) |
| $f_{12}$ | 11.0551(421.2096) | 1.9596e+03(3.1327e+07) | 0.3979(0) | 0.3979(0) |
| $f_{13}$ | 0(0) | 0(0) | 0.2750(0.0084) | 0.3944(0.0121) |
| $f_{14}$ | 6.6771e-12(1.0702e-21) | 6.6946e-09(1.0756e-15) | 5.0148(185.7615) | 80.0579(33.3354) |
| $f_{15}$ | 0(0) | 1.4020e-13(4.7177e-25) | 0.4123(1.6140e-13) | 0.4123(1.4388e-27) |
| $f_{16}$ | 0(0) | 0(0) | -1.1504(1.5460e-12) | -1.1504(3.3189e-27) |
| $f_{17}$ | 0.9980(5.3883e-21) | 0.9980(0) | 0.9980(5.3883e-21) | 0.9980(0) |
| $f_{18}$ | -9.6003e-04(3.1558e-10) | 0.0044(1.0963e-10) | 9.6003e-04(3.1558e-10) | 0.0044(1.0963e-10) |
| $f_{19}$ | -1.0316(7.2889e-16) | -1.0316(1.6763e-31) | -1.0316(7.2889e-16) | -1.0316(1.6763e-31) |
| $f_{20}$ | 0.3979(6.3308e-11) | 0.3979(0) | 0.3979(6.3308e-11) | 0.3979(0) |
| $f_{21}$ | 3.0000(6.0843e-19) | 3.0000(3.1554e-30) | -3.3215(8.4559e-12) | -3.3215(1.9722e-31) |
| $f_{22}$ | -10.1532(6.6789e-13) | -10.1532(1.2204e-19) | -10.1532(6.6789e-13) | -10.1532(1.2204e-19) |
| $f_{23}$ | -10.4029(5.2199e-09) | -10.4029(3.8379e-15) | -10.4029(5.2199e-09) | -10.4029(3.8379e-15) |
Table 8. Means and variances under different algorithms for \( D = 20 \) and \( N = 30 \) (25 runs)

| Function | \( f_1 \) | \( f_2 \) | \( f_3 \) | \( f_4 \) | \( f_5 \) |
|----------|----------|----------|----------|----------|----------|
| NFO-resample | 5.0244e-38 (6.0588e-74) | 2.2622e-23 (1.2282e-44) | 1.3053e-07 (4.0837e-13) | 6.3508e-154 (9.6797e-306) | 1.528e+04 (5.277e+187) |
| PSO, cf._local | 3.1842 (5.2344) | 1.5070 (0.5649) | 12.8833 (67.5186) | 1.791e+46 (3.862e+91) | 1.528e+04 (5.277e+187) |
| UPSO | 2.0381e-15 (8.466e-30) | 3.755e-11 (2.915e-20) | 1.7022 (2.5477) | 2.806e+04 (9.112e+18) | 1.528e+04 (5.277e+187) |
| FDRPSO | 3.0638e-09 (3.2419e-18) | 8.123e-06 (8.022e-12) | 1.9813 (3.5545) | 1.436e+05 (4.832e+17) | 1.528e+04 (5.277e+187) |
| CLPSO | 3.1939e-06 (5.927e-12) | 1.5475e-04 (1.827e-09) | 786.7243 (4.326e+04) | 2.099e+06 (5.695e+131) | 1.528e+04 (5.277e+187) |
| LDWPSO | 1.2282 (0.9361) | 0.3676 (0.0501) | 49.3145 (9.313e+03) | 1.7202 (2.2547) | 1.096e+02 (2.634e+18) |
| DE | 99.0340 (1.033e+05) | 1.6477 (6.3922) | 625.9111 (9.983e+05) | 1.7202 (2.2547) | 1.096e+02 (2.634e+18) |
| Function | \( f_6 \) | \( f_7 \) | \( f_8 \) | \( f_9 \) | \( f_{10} \) |
|----------|----------|----------|----------|----------|----------|
| NFO-resample | 0.4585 | 15.4349 (4.3585) | 1.2137e+03 (1.0794e+06) | 41.7600 (706.5824) | 0.4164 (0.0697) |
| PSO, cf._local | 4.7672 (1.456e+04) | 0.5480 (0.0865) | 31.3346 (2.283e+09) | 1.7202 (2.2547) | 1.096e+02 (2.634e+18) |
| UPSO | 87.8288 (1.494e+03) | 0.6045 (0.1044) | 14.1200 (36.3456) | 0.4455 (0.0878) | 29.9535 (6.8640) |
| LDWPSO | 3.3685e-06 (1.422e-14) | 2428.00 (706.5824) | 0.2584 (0.0037) | 50.3008 (1.757e+07) | 0.4164 (0.0697) |
| DE | 3.3685e+06 (1.422e+14) | 2428.00 (706.5824) | 0.2584 (0.0037) | 50.3008 (1.757e+07) | 0.4164 (0.0697) |
| Function | \( f_{11} \) | \( f_{12} \) | \( f_{13} \) | \( f_{14} \) | \( f_{15} \) |
|----------|----------|----------|----------|----------|----------|
| NFO-resample | 0.0084 (1.731e-04) | 1.6987 (5.8474) | 0.0084 (1.731e-04) | 1.6987 (5.8474) | -0.3243 (1.7407) |
| FDRPSO | 1.6912e-05 (3.062e-11) | 0.4227 (1.814e-20) | -1.1509 (6.435e-06) | 0.4164 (0.0697) | 29.9535 (6.8640) |
| CLPSO | 0.0011 (1.455e-07) | 0.8247 (7.186e-16) | -1.504 (3.925e-12) | 0.4164 (0.0697) | 29.9535 (6.8640) |
| LDWPSO | 2.7815 (0.4828) | 0.9390 (0.0336) | 3.0484 (4.9592) | 0.2584 (0.0037) | 50.3008 (1.757e+07) |
| DE | 1.0774 (1.0152) | 1.5132 (1.0152) | 1.0316 (1.025e-31) | 3.0484 (4.9592) | 0.2584 (0.0037) |
| Function | \( f_{16} \) | \( f_{17} \) | \( f_{18} \) | \( f_{19} \) | \( f_{20} \) |
|----------|----------|----------|----------|----------|----------|
| NFO-resample | 3.0000 (1.150e+04) | -3.2793 (1.003e-12) | -8.4721 (3.347e-12) | -8.3865 (1.353e-12) | 0.3979 (0.0044) |
| PSO, cf._local | 3.0000 (1.508e+09) | -3.3215 (1.546e-10) | -10.1352 (5.141e-17) | -10.4029 (2.929e+17) | 0.3979 (0.0044) |
| UPSO | 3.0000 (1.508e+09) | -3.3215 (1.546e-10) | -10.1352 (5.141e-17) | -10.4029 (2.929e+17) | 0.3979 (0.0044) |
| FDRPSO | 3.0000 (1.508e+09) | -3.3215 (1.546e-10) | -10.1352 (5.141e-17) | -10.4029 (2.929e+17) | 0.3979 (0.0044) |
| CLPSO | 3.0000 (6.053e-29) | -3.2741 (1.003e-12) | -8.4721 (3.347e-12) | -8.3865 (1.353e-12) | 0.3979 (0.0044) |
| LDWPSO | 3.0000 (6.053e-29) | -3.2741 (1.003e-12) | -8.4721 (3.347e-12) | -8.3865 (1.353e-12) | 0.3979 (0.0044) |
Table 9. Convergence steps under the eight algorithms with $D = 20, N = 30$

| Function  | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------|
| PSO-resample | 432   | 537   | 794   | 75    | –     | 348   | 87    | –     | 648   | 746     |
| NFO-resample | 252   | 237   | 54    | 35    | 509   | 182   | 53    | 85    | 257   | 292     |
| PSO_{cf..local} | –     | 1026  | –     | 76    | –     | 357   | 361   | –     | 1059  | 1145    |
| UPSO       | 706   | 745   | –     | 63    | –     | 519   | 1700  | 1230  | 877   | 877     |
| FDRPSO     | 1579  | 1712  | –     | 116   | –     | 1294  | 735   | 137   | 1737  | –       |
| CLPSO      | 1638  | 1654  | –     | 133   | –     | 1201  | 820   | –     | 1921  | 1888    |
| LDWPSO     | 1050  | 930   | 1296  | 171   | 1468  | 676   | 1092  | 1068  | 919   | 973     |
| DE         | 17    | 25    | 38    | 28    | 19    | 500   | 19    | 41    | 49    |          |

| Function  | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ | $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{20}$ |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| PSO-resample | –         | –         | 148       | 44        | 46        | 55        | 97        | 107       | 93        |           |
| NFO-resample | 309     | 249      | 41        | 25        | 17        | 39        | 39        | 73        | 74        | 72        |
| PSO_{cf..local} | –         | 1408      | 86        | 158       | 52        | 53        | 65        | 85        | 105       | 119       |
| UPSO       | 530      | 810      | 204       | 138       | 44        | 52        | 58        | 89        | 122       | 121       |
| FDRPSO     | 1289     | 1550     | 203       | 152       | 106       | 126       | 162       | 182       | 538       | 552       |
| CLPSO      | 1358     | 1617     | 263       | 351       | 144       | 200       | 348       | 586       | 861       | 737       |
| LDWPSO     | 1077     | 1072     | 169       | 208       | 82        | 72        | 110       | 179       | 278       | 286       |
| DE         | 29       | 18       | 19        | 22        | 17        | 32        | 22        | 25        | 27        | 35        |

Figure 4 and Table 9 depict the convergence performance of different test functions with $D = 20, N = 30$ under eight algorithms. NFO-resample is in the top two on all function except $f_7, f_{15}$. For $f_7, f_{15}$, NFO-resample converges the fastest. PSO-resample converges the faster on function $f_1 \sim f_4, f_6, f_7, f_9, f_{10}, f_{13} \sim f_{20}$ and it is the third fastest on $f_1 \sim f_3, f_6, f_7, f_9, f_{10}, f_{15} \sim f_{17}, f_{20}$. It is noted that, the convergence speed of DE is the fastest. Yet the solution of DE from Table 8 is not as good as those of PSO-resample and NFO-resample. On the whole, we can conclude that the PSO-resample and NFO-resample are superior algorithms in terms of the convergence performance.

5. Conclusion. This paper proposes a new technique to improve two intelligent algorithms. One is the global search algorithm, the particle swarm optimization, and the other, the neighborhood field optimization, utilizes the local information with the principle of “learning from neighbors”. This technique is to introduce the re-sampling step of particle filter followed by particle variation in the later searching stage of PSO and NFO. We define a weight of a particle according to the distance from this particle to the current best particle, then the re-sampling step of particle filter can be borrowed to copy particles with high weights and lessen particles with low weights. With this method, the local search ability is enhanced greatly and the searching efficiency of the algorithm is elevated accordingly. With this method, the local search ability, which is more emphasized during the later searching stage, is enhanced greatly and the searching efficiency of the algorithm is elevated accordingly. And to avert the identity of particles, the existing method of particle variation is executed after the re-sampling step. Extensive simulation experiments are performed to compare the proposed PSO-resample and NFO-resample with the standard PSO and NFO and other metaheuristic algorithms on both uni-modal and multi-modal test functions. The results show that PSO-resample and NFO-resample can achieve significantly better performance respectively, in terms of the
Figure 4. Comparison of the convergence rate under eight algorithms ($D = 20, N = 30$)
accuracy of the solution, the global search ability, the stability and robustness, and the convergence speed. Based on our results, it is conjectured that the proposed technique can be general. This arises from the observation that this technique can be applied to the later searching stage of other particle-based intelligent algorithms in a similar way. Nevertheless, it is fairly promising to examine the technique for those algorithms, which can possibly improve their performance.

Acknowledgments. The authors would like to thank Dr. Zhou Wu from Chongqing University for his valuable comments. This research was supported in part by the National Natural Science Foundation of China (61304090), the Department of Science & Technology of Liaoning Province (20150183), the Department of Education of Liaoning Province (L2013132) and Innovation Research Funds of Liaoning Technical University (2013031T).

REFERENCES

[1] K. Amin and M. Guerrero-Zapata, A hybrid multiobjective RBF-PSO method for mitigating dos attacks in named data networking, Neurocomputing, 151 (2015), 1262–1282.
[2] K. Amin and M. Guerrero-Zapata, A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks, Neurocomputing, 149 (2015), 1253–1269.
[3] M. S. Arulampalam, S. Maskell, N. Gordon and T. clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Transactions on Signal Processing, 50 (2002), 174–188.
[4] T. M. Blackwell and P. Bentley, Don’T push me! Collision-avoiding swarms, Proceedings of the 2002 Congress on Evolutionary Computation, 2 (2002), 1691–1696.
[5] J. Carpenter, P. Clifford and F. Fearnhead, An improved particle filter for non-linear problems, IEE Proceedings-Radar, Sonar and Navigation, 146 (1999), 2–7.
[6] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 6 (2002), 58–73.
[7] D. Crisan and A. Doucet, A survey of convergence result on particle filtering methods for practitioners, IEEE Transactions on Signal Processing, 50 (2002), 736–746.
[8] A. Doucet, S. Godsill and C. Andrieu, On sequential Monte Carlo sampling method for Bayesian filtering, Statistics and Computing, 10 (2000), 197–208.
[9] A. Doucet, J. Freitas and N. Gordon, Sequential Monte Carlo Methods in Practice, Springer-Verlag, 2001.
[10] R. C. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the 6th International Symposium on Micro machine and Human Science, (1995), 39–43.
[11] F. Glover, Tabu search-part II, ORSA Journal on Computing, 2 (1990), 4–32.
[12] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.
[13] N. J. Gordon, D. J. Salmond and A. F. M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, Radar and Signal Processing, IEE Proceedings F, 140 (1993), 107–113.
[14] R. Greiner, POLO: A probabilistic hill-climbing algorithm, Artificial Intelligence, 84 (1996), 177–208.
[15] J. Grobler, A. P. Engelbrecht, G. Kendall and S. Yadavalli, Heuristic space diversity control for improved meta-hyper-heuristic performance, Information Sciences, 300 (2015), 49–62.
[16] J. Grobler and A. P. Engelbrecht, Headless chicken particle swarm optimization algorithms, Tan Y., Shi Y., Niu B. (eds) Advances in Swarm Intelligence. ICSI 2016. Lecture Notes in Computer Science, 9712 (2016), 350–357.
[17] J. Grobler and A. P. Engelbrecht, A scalability analysis of particle swarm optimization roaming behaviour, Advances in Swarm Intelligence. ICSI 2017. Lecture Notes in Computer Science, 10385 (2017), 119–130.
[18] J. D. Hol, T. B. Schon and F. Gustafsson, On resampling algorithms for particle filters, IEEE Nonlinear Statistical Signal Processing Workshop, (2006), 79–82.
[19] B. J. Jain, H. Pohlheim and W. Joachim, On termination criteria of evolutionary algorithms, Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, (2001), 768–775.

[20] J. Kennedy and R. C. Eberhart, Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, (1995), 1942–1948.

[21] J. Kennedy and R. Mendes, Population structure and particle swarm performance, Proceedings of the 2002 Congress on Evolutionary Computation, 2 (2002), 1671–1676.

[22] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, Science, 220 (1983), 671–680.

[23] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, Journal of Computational and Graphical Statistics, 5 (1996), 1–25.

[24] J. Lampinen and R. Storn, Differential evolution, Onwubolu G, Babu BV (eds) New Optimization Techniques in Engineering, (2004), 123–166.

[25] 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, 10 (2006), 281–295.

[26] J. S. Liu and R. Chen, Sequential Monte Carlo methods for dynamic systems, Journal of the American Statistical Association, 93 (1998), 1032–1044.

[27] R. Mendes, J. Kennedy and J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation, 8 (2004), 204–210.

[28] K. E. Parsopoulos and M. N. Vrahatis, UPSO–A unified particle swarm optimization scheme, Lecture Series on Computational Sciences, 1 (2004), 868–873.

[29] T. Peram, K. Veeramachaneni and C. K. Mohan, Fitness-distance-ratio based particle swarm optimization, Proceedings of the 2003 IEEE on Swarm Intelligence Symposium, (2003), 174–181.

[30] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, (1998), 69–73.

[31] R. Storn and K. Price, Differential evolutional simple and efficient heuristic for global optimization over continuous space, Journal of Global Optimization, 11 (1997), 341–359.

[32] P. N. Suganthan, Particle swarm optimizer with neighborhood operator, Proceedings of the 1999 Congress on Evolutionary Computation, 3 (1999), 1958–1962.

[33] M. D. Vose, Simple Genetic Algorithm: Foundation and Theory, MIT Press, MI, 1999.

[34] Z. Wu and T. W. S. Chow, A local multiobjective optimization algorithm using neighborhood field, Structural and Multidisciplinary Optimization, 46 (2012), 853–870.

[35] Z. Wu and T. W. S. Chow, Neighborhood field for cooperative optimization, Soft Computing, 17 (2013), 819–834.

[36] Z. Wu and T. W. S. Chow, Binary neighborhood field optimization for unit commitment problems, IET Generation Transmission and Distribution, 7 (2013), 299–308.

[37] C. Yang, L. Gu and W. Gui, Particle swarm optimization algorithm with adaptive mutation, Computer Engineering, 34 (2008), 188–190.

[38] X. Yao, Y. Liu and G. M. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation, 3 (1999), 82–102.

[39] B. Yao, B. Yu, P. Hu, J. Gao and M. H. Zhang, An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot, Annals of Operations Research, 242 (2016), 303–320.

[40] T. T. Zhao, Q. F. Cheng and Z. F. Wang, Nonlinear model predictive control optimization with improved particle swarm algorithm, Liaoning Gongcheng Jishu Daxue Xuebao (Ziran Kexue Ban)/Journal of Liaoning Technical University (Natural Science Edition), 34 (2015), 517–522.

Received May 2017; revised October 2017.

E-mail address: chengqifeng@tsinghua.org.cn (Qifeng Cheng)
E-mail address: hanxue.boda@163.com (Xue Han)
E-mail address: happyzhaotingting@qq.com (Tingting Zhao)
E-mail address: sarma.yadavalli@up.ac.za (V S Sarma Yadavalli)