Research Article

Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems

Martins Akugbe Arasomwan and Aderemi Oluyinka Adewumi

*School of Mathematics, Statistics, and Computer Science, University of Kwazulu-Natal South Africa,
Private Bag X54001, Durban 4000, South Africa*

Correspondence should be addressed to Aderemi Oluyinka Adewumi; laremtj@gmail.com

Received 31 October 2013; Accepted 29 December 2013; Published 25 February 2014

Academic Editors: T. Chen, Q. Cheng, and J. Yang

Copyright © 2014 M. A. Arasomwan and A. O. Adewumi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new local search technique is proposed and used to improve the performance of particle swarm optimization algorithms by addressing the problem of premature convergence. In the proposed local search technique, a potential particle position in the solution search space is collectively constructed by a number of randomly selected particles in the swarm. The number of times the selection is made varies with the dimension of the optimization problem and each selected particle donates the value in the location of its randomly selected dimension from its personal best. After constructing the potential particle position, some local search is done around its neighbourhood in comparison with the current swarm global best position. It is then used to replace the global best particle position if it is found to be better; otherwise no replacement is made. Using some well-studied benchmark problems with low and high dimensions, numerical simulations were used to validate the performance of the improved algorithms. Comparisons were made with four different PSO variants, two of the variants implement different local search technique while the other two do not. Results show that the improved algorithms could obtain better quality solution while demonstrating better convergence velocity and precision, stability, robustness, and global-local search ability than the competing variants.

1. Introduction

Optimization comes to focus when there are needs to plan, take decisions, operate and control systems, design models, or seek optimal solutions to varieties of problems faced from day to day by different people. A number of these problems, which can be formulated as continuous optimization problems, are often approached with limited resources. Dealing with such problems, most especially when they are large scale and complex, has attracted the development of different nature-inspired optimization algorithms. These algorithms display problem-solving capabilities for researchers to solve complex and challenging optimization problems with many success stories. Swarm-based techniques are a family of nature-inspired algorithms and are population-based in nature; they are also known as evolutionary computation techniques. Particle swarm optimization (PSO) technique is a member of swarm-based techniques which is capable of producing low cost, fast, and robust solutions to several complex

optimization problems. It is a stochastic, self-adaptive, and problem-independent optimization technique and was originally proposed in 1995 by Eberhart and Kennedy as simulation of a flock of bird or the sociological behavior of a group of people [1, 2]. From the time this concept was brought into optimization, it has been used extensively in many fields which include function optimization and many difficult real-world optimization problems [3–5].

PSO technique was initially implemented with few lines of codes using basic mathematical operations with no major adjustment needed to adapt it to new problems and it was almost independent of the initialization of the swarm [6]. It needs few parameters to operate with for successful and efficient behavior in order to obtain quality solutions. To implement this technique, a number of particles, which are characterized by positions and velocities, called swarm are required to be randomly distributed in a solution search space depending on the boundaries defined for the design variables of the problem being optimized. The number of

design variables determines the dimensionality of the search space. If d -dimensional space is considered, the position and velocity of each particle are represented as the vectors $X_i = (x_{i1}, \dots, x_{id})$ and $V_i = (v_{i1}, \dots, v_{id})$, respectively. Every particle has a memory of its personal experience which is communicated to all reachable neighbours in the search space to guide the direction of movement of the swarm. Also, the quality of each particle (solution) is determined by the objective function of the problem being optimized and the particle with best quality is taken as the global solution towards which other particles will converge. The common practice is for the technique to maintain a single swarm of particles throughout its operation. This process of seeking optimal solution involves the adjustments of the position and velocity of each particle in each iteration using

$$V_i(t+1) = \omega V_i(t) + \text{coeff}_1 (P_i - X_i) + \text{coeff}_2 (P_g - X_i), \quad (1)$$

$$X_i(t+1) = X(t) + V_i(t+1). \quad (2)$$

In (1), P_i and P_g are vectors representing the i th particle personal best and swarm global best positions, respectively; $\text{coeff}_1 = c_1 r_1$ and $\text{coeff}_2 = c_2 r_2$; c_1 and c_2 are acceleration factors known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of P_i and P_g ; r_1 and r_2 are random numbers between 0 and 1; t is iteration index. The symbol ω is the inertia weight parameter which was introduced into the original PSO in [7]. The purpose of its introduction was to help the PSO algorithm balance its global and local search activities.

There are possibilities of the positions and velocities of the particles in the swarm increasing in value beyond necessary when they are updated. As a measure, the positions are clamped in each dimension to the search range $[X_{\min}, X_{\max}]$ of the design variables, where $X_{\min}$ and $X_{\max}$ represent the lower and upper bounds of a particle's position, respectively, while their velocities are controlled to be within a specified range $[V_{\min}, V_{\max}]$, where $V_{\min}$ and $V_{\max}$ represent the lower and upper bounds of a particle's velocity, respectively. The idea of velocity clamping which was introduced by [1, 2, 8] and extensively experimented with in [9] has led to significant improvement as regards the performance of PSO. This is so because the particles could concentrate, taking reasonably sized steps to search through the search space rather than bouncing about excessively. A major feature that characterizes an efficient optimization algorithm is the ability to strike a balance between local and global search. Global search involves the particles being able to advance from a solution to other parts of the search space and locate other promising candidates while local search means that the particle is capable of exploiting the neighbourhood of the present solution for other promising candidates. In PSO, as the rate of information sharing increases among the particles they migrate towards the same direction and region in the search space. If any of the particles could not locate any better global solution after some time, they will eventually converge about the existing one which may not be the global minimum due to lack of exploration power; this is known

as premature convergence. This type of behaviour is more likely when the swarm of particles is overconcentrated. It could also occur when the optimization problem is of high dimension and/or nonconvex. One of the possible ways to prevent this premature convergence is to embed a local search technique into PSO algorithm to help improve the quality of each solution by searching its neighbourhood. After the improvement, better information is communicated among the particles thereby increasing the algorithm's ability to locate better global solution in course of optimization. Hill climbing, modified Hooke and Jeeves, gradient descent, golden ratio, Stochastic local search, adaptive local search, local interpolation, simulated annealing, and chaotic local search are different local search techniques that have been combined with PSO to improve its local search ability [10–18].

In this paper, a different local search technique was proposed to harness the global search ability of PSO and improve on its local search efforts. This technique is based on the collective efforts of randomly selected (with replacement) particles a number of times equal to the size of the problem dimension. When a particle is selected, it is made to contribute the value in the position of its randomly selected dimension from its personal best. The contributed values are then used to form a potential global best solution which is further refined. This concept could offer PSO the ability to enhance its performance in terms of convergence speed, local search ability, robustness, and increased solution accuracy. The local search technique was hybridized with two of the existing PSO variants, namely, random inertia weight PSO (RIW-PSO) and linear decreasing inertia weight PSO (LDIW-PSO), to form two new variants. Numerical simulations were performed to validate the efficiencies of each of them and some statistical analyses were performed to ascertain any statistically significant difference in performance between the proposed variants and the old ones. From the results obtained it was shown that the proposed variants are very efficient.

In the sections that follow, RIW-PSO and LDIW-PSO are briefly described in Section 2; the motivation and description of the proposed local search technique are presented in Section 3 while the improved PSO with local search technique is described in Section 4. Numerical simulations are performed in Section 5 and Section 6 concludes the paper.

2. The Particle Swarm Optimization Variants Used

Two PSO variants were used to validate the proposed improvement of the performance of PSO technique. The variants are LDIW-PSO and RIW-PSO. These were chosen because of the evidence available in the literature that they are less efficient in optimizing many continuous optimization problems [19–21]. These variants are succinctly described below.

2.1. PSO Based on Linear Decreasing Inertia Weight (LDIW-PSO). This variant was proposed in [9] after the inertia

weight parameter was introduced into the original PSO by [7]. It implements the linear decreasing inertia weight strategy represented in (3) which decreases from some high value which facilitates exploration to a low value which on the other hand promotes exploitation. This greatly improved the performance of PSO. LDIW-PSO does global search at the beginning and converges quickly towards optimal positions but lacks exploitation power [9] and the ability required to jump out of the local minimum most especially when being in the multimodal landscape. Some improvements on LDIW-PSO exist in the literature [6, 9, 22]:

$$\omega_i = (\omega_{\text{start}} - \omega_{\text{stop}}) \left(\frac{\text{MAX}_{\text{itr}} - i}{\text{MAX}_{\text{itr}}} \right) + \omega_{\text{stop}}, \quad (3)$$

where ω_{start} and ω_{stop} are the initial and final values of inertia weight, i is the current iteration number, MAX_{itr} is the maximum iteration number, and $\omega_i \in [0, 1]$ is the inertia weight value in the i th iteration. Apart from the problem of premature convergence, this variant was found inefficient in tracking a nonlinear dynamic system because of the difficulty in predicting whether exploration (a larger inertia weight value) or exploitation (a smaller inertia weight) will be better at any given time in the search space of the nonlinear dynamic system [23].

2.2. PSO Based on Random Inertia Weight (RIW-PSO). Due to the improved performance of PSO when the constant inertia weight was introduced into it [7], a new era of research was indirectly initiated and this has attracted the attentions of many researchers in the field. The inefficiency of linear decreasing inertia weight, which linearly decreases from 0.9 to 0.4, in tracking a nonlinear dynamic system prompted the introduction of RIW which randomly varies within the same range of values. Random adjustment is one of the strategies that have been proposed to determine the inertia weight value to further improve on the performance of PSO. This strategy is nonfeedback in nature and the inertia weight takes different value randomly at each iteration, from a specified interval. In line with this, random inertia weight strategy represented in (4) was introduced into PSO by [23] to enable the algorithm track and optimize dynamic systems. In the equation, $\text{rand}()$ is a uniform random number in the interval $[0, 1]$ which make the formula generate a number randomly varying between 0.5 and 1.0, with a mean value of 0.75. When c_1 and c_2 are set to 1.494, the algorithm seems to demonstrate better optimizing efficiency. The motivation behind the selection of these values was Clerc's constriction factor [23]:

$$\omega = 0.5 + \frac{\text{rand}()}{2}. \quad (4)$$

Not much is recorded in the literature regarding the implementation of this variant of PSO. Some of the few implementations found in the literature are recorded in [6, 20–22].

3. Proposed Local Search Technique

The basic principle underlying the optimizing strategy of PSO technique is that each particle in the swarm communicates

their discoveries to their neighbours and the particle with the best discovery attracts others. While this strategy looks very promising, there is the risk of the particles being susceptible to premature convergence, especially when the problem to be optimized is multimodal and high in dimensionality. The reason is that the more the particles share their discoveries among themselves, the higher their identical behaviour is until they converge to the same area in the solution search space. If none of the particle could discover better global best, after some time all the particles will converge about the existing global best which may not be the global minimizer.

One of the motivations for this local search technique is the challenge of premature convergence associated with PSO technique which affects its reliability and efficiency. Another motivation is the decision-making strategy used by the swarm in searching for optimal solution to optimization problems. The decision is dictated by a single particle in the swarm; that is, other particles follow the best particle among them to search for better solution. Involving more than one particle in the decision making could lead to a promising region in the search space where optimal solution could be obtained.

The description of the local search technique is as follows: after all the particles have obtained their various personal best positions, each particle has an equal chance of being selected to contribute its idea towards how a potential location in the search space where better global best could be obtained. As a result, a number of particles equal to the dimension of the problem being optimized are randomly selected (with replacement). Each selected particle contributes an idea by donating the value in the location of its randomly selected dimension from its personal best. All the ideas contributed by the selected particles are collectively used (hybridized) to construct a potential solution in the solution search space. After constructing the potential solution, some searches are locally done around its neighbourhood with the hope of locating a better solution in comparison with the current global solution. If a better solution is found, it is then used to replace the current global solution; otherwise no replacement is made.

In this local search, the potential new position is denoted by $\vec{y}$ and is sampled from the neighbourhood of the collectively constructed potential global solution represented as $\vec{P}$ by

$$\vec{y} \leftarrow \vec{P} + \vec{a}, \quad (5)$$

where $\vec{a} \sim U[-\vec{r}, \vec{r}]$ is a random vector picked uniformly from the range $[-\vec{r}, \vec{r}]$ and $\vec{r}$ is the search radius which is initially set to $\text{max}R$ (maximum radius for local search). The local search technique moves from position $\vec{P}$ to position $\vec{y}$ when there is improvement to the fitness. If there is no improvement on the fitness of $\vec{P}$ by $\vec{y}$, the search radius is linearly decreased by multiplying it with a factor q using

$$\vec{r} \leftarrow q \times \vec{r}, \quad (6)$$

$$q \leftarrow (\text{max}R - \text{min}R) \times \frac{t}{\text{max}T} + \text{min}R,$$

```

 $\vec{r}, \vec{a}, \vec{y} \leftarrow$  new arrays, each of length Dim
 $\vec{r} \leftarrow \max R$ 
 $t \leftarrow 0$ 
While ( $t < \max T$ ) do
   $t \leftarrow t + 1$ 
   $\vec{a} \leftarrow U(-\vec{r}, \vec{r})$ 
  for  $j \leftarrow 1$  to problem Dimension
    randomly select any particle  $i$ 
    randomly select a dimension  $d$  from the personal best  $P$  of the selected particle  $i$ 
     $\vec{y}^j \leftarrow \vec{P}_i^d + \vec{a}^j$ 
  end for
  validate for search space boundary
  If  $f(\vec{y}) < gFit$ 
     $gPos \leftarrow \vec{y}$ 
     $gFit \leftarrow f(\vec{y})$ 
  else
     $q \leftarrow (\max R - \min R) \times \frac{t}{\max T} + \min R$ 
     $\vec{r} \leftarrow q \times \vec{r}$ 
  end if
end while
Return  $gPos$  and  $gFit$ 

```

ALGORITHM 1: Collective local unimodal search.

Begin PSO_{CLUS} Algorithm**Step 1. Definition Phase**

- (1.1) function to optimize as f
- (1.2) Parameter
 - (1.2.1) swarm size
 - (1.2.2) problem dimension
 - (1.2.3) solution search space
 - (1.2.4) particle velocity range

Step 2. Initialized phase**For all particles randomly initialized in search space**

- (2.1) position $x_i \leftarrow (x_{i1}, \dots, x_{id})$
- (2.2) velocity $v_i \leftarrow (v_{i1}, \dots, v_{id})$,
- (2.3) $pbest_i \leftarrow (x_{i1}, \dots, x_{id})$
- (2.4) $gbest \leftarrow$ best of $pbest_i$
- (2.5) evaluate $f(x_i)$ using objective function of problem

Step 3. Operation Phase**Repeat until a stopping criterion is satisfied**

- (3.1). Compute inertia weight using any inertia weight formula
- (3.2). For each particle i
 - (3.2.1). update v_i for particle using (1)
 - (3.2.2). validate for velocity boundaries
 - (3.2.3). update x_i for particle using (2)
 - (3.2.4). validate for position boundaries
 - (3.2.5). If $f(x_i) < f(pbest_i)$ then $pbest_i \leftarrow x_i$
- (3.3). $gbest \leftarrow$ best of $pbest_i$
- (3.4). Implement local search using CLUS in Algorithm 1

Step 4. Solution Phase

- (4.1). $x^* \leftarrow gbest$
- (4.2). $f^* \leftarrow f(gbest)$
- (4.3). Return x^* and f^*

End PSO_{CLUS} AlgorithmALGORITHM 2: Algorithm for PSO_{CLUS}.

TABLE 1: Parameter settings for experiment.

Parameter	$\omega_{\min}$	$\omega_{\max}$	$c_1 = c_2$	$V_{\min}$	$V_{\max}$	minR	maxR	maxT
Value	0.9	0.4	1.494	$0.05 * X_{\min}$	$0.05 * X_{\max}$	0.01	2.0	100

where $\max T$ is the maximum number of times the neighbourhood of $\vec{P}$ is to be sampled, t is the current time the neighbourhood is being sampled, and $\min R$ is the minimum radius for the local search.

This proposed local search technique has been named collective local unimodal search (CLUS) technique. It has some trace of similarity in operation with local unimodal sampling (LUS) technique [24]. But they are quite different in the sense that, while LUS randomly picks a potential solution from the entire population, CLUS constructs a potential solution using the collective efforts of a randomly selected number of particles from the swarm. Also, CLUS uses a linear method to decrease the search radius (step size) in the neighbourhood of the potential solution which is different from the method applied by LUS during optimization. The CLUS technique is presented in Algorithm 1. In the technique, $gFit$ and $\vec{gPos}$ represent the current global fitness value and its corresponding position in the search space.

4. Improved PSO with Collective Unimodal Local Search (PSO_{CLUS})

The RIW-PSO increases convergence in early iterations and does more of global search activities but soon gets stuck in local optima because of lack of local search ability. Also, LDIW-PSO does global search at earlier part of its iteration but lacks enough momentum to do local search as it gets towards its terminal point of execution. The aim of this paper is to make a general improvement on the performance of PSO which can be applied to any of its variants. To achieve this, the two PSO variants were hybridized with the proposed collective local unimodal search (CLUS) technique which takes advantage of their global search abilities to do some neighbourhood search for better results. The improved PSO algorithm is presented in Algorithm 2.

5. Numerical Simulations

In this section, the improved algorithm (PSO_{CLUS}) was implemented using the inertia weight strategy of RIW-PSO and the variant was labeled R -PSO_{CLUS}. It was also implemented using the inertia weight strategy of LDIW-PSO and the variant was labeled L -PSO_{CLUS}. The performances of R -PSO_{CLUS} and L -PSO_{CLUS} were experimentally tested against those of RIW-PSO and LDIW-PSO, respectively. The maximum number of iterations allowed was 1000 for problems with dimensions less than or equal to 10, 2000 for 20-dimensional problems, and 3000 for 30-dimensional problems. A swarm size of 20 was used in all the experiments and twenty-five independent runs were conducted to collect data for analysis. The termination criteria for all the algorithms were set to be as maximum number of iterations relative to the

problems' dimensions. A run, in which an algorithm is able to satisfy the set success criteria (see Table 1) before or at the maximum iteration, is considered to be successful. To further prove the efficiency of the proposed local search technique, the proposed PSO variants were also compared with some existing PSO variants hybridized with different local search techniques. They are PSO with golden ratio local search [15] and PSO with local interpolation search [18]. A total of 6 different experiments were conducted.

- (i) R -PSO_{CLUS} was compared with PSO with golden ratio local search (GLSPSO);
- (ii) R -PSO_{CLUS} was compared with PSO with local interpolation search (PSO_{lis});
- (iii) R -PSO_{CLUS} was compared with RIW-PSO;
- (iv) L -PSO_{CLUS} was compared with PSO with golden ratio local search (GLSPSO);
- (v) L -PSO_{CLUS} was compared with PSO with local interpolation search (PSO_{lis});
- (vi) L -PSO_{CLUS} was compared with LDIW-PSO.

The application software was developed in Microsoft Visual C# programming language.

5.1. Test Problems. A total of 21 problems were used in the experiments. These problems have different degrees of complexity and multimodality which represents diverse landscapes enough to cover many of the problems which can arise in global optimization problems. Shown in Table 2 are the problems dimensions, optimal fitness values, and success thresholds. Presented in Table 3 are the definitions, characteristics (US: unimodal separable, UN: unimodal non-separable, MS: multimodal separable, and MN: multimodal nonseparable), and search ranges of the problems. More details on the benchmark problems can be found in [22, 25–27].

5.2. Parameter Setting. The additional parameters that were set in the experiment are inertia weight threshold for LDIW-PSO ($\omega_{\min}$ and $\omega_{\max}$), acceleration coefficients (c_1 and c_2), velocity thresholds ($V_{\min}$ and $V_{\max}$), minimum radius ($\min R$), and maximum radius ($\max R$) for local search as well as the maximum number of neighbourhood sampling ($\max T$) during the local search. The respective settings of these parameters are shown in Table 1. The parameters r_1 and r_2 were randomly generated using the uniform random number generator. The values of $\omega_{\min}$ and $\omega_{\max}$ were chosen for LDIW-PSO based on the experiments conducted in [9]; values for c_1 and c_2 were chosen for RIW-PSO based on the recommendation in [23] and it was also used for LDIW-PSO because it was discovered in course of the experiments in this paper that these values make LDIW-PSO perform better than

TABLE 2: Benchmark problems.

Number	Problem	Dimensions	Optimal value	Success threshold
1	Ackley	10, 20, 30	0	10^{-5}
2	Booth	2	0	10^{-5}
3	Easom	2	-1	-1
4	Griewank	10, 20, 30	0	10^{-5}
5	Dixon-Price	10, 20, 30	0	10^{-5}
6	Levy	10, 20, 30	0	10^{-5}
7	Michalewicz	5	-4.687	-4.687
8	Noisy Quartic	10, 20, 30	0	10^{-5}
9	Noncontinuous Rastrigin	10, 20, 30	0	20
10	Rastrigin	10, 20, 30	0	20
11	Rosenbrock	10, 20, 30	0	20
12	Rotated Ellipsoid	10, 20, 30	0	10^{-5}
13	Salomon	5	0	10^{-5}
14	Schaffer's f6	2	0	10^{-5}
15	Schwefel	10, 20, 30		
16	Schwefel P2.22	10, 20, 30	0	10^{-5}
17	Shubert	2	-186.7309	-186.7309
18	Sphere	10, 20, 30	0	10^{-5}
19	Step	10, 20, 30	0	10^{-5}
20	Sum Squares	10, 20, 30	0	10^{-5}
21	Trid	6	-50	-50

the commonly used value of 2.0. The settings for $V_{\min}$ and $V_{\max}$ were done based on the outcome of experimental studies in [8].

5.3. Performance Measurement. The efficiency of the algorithms was tested against the set of benchmark problems given in Table 2 and numerical results obtained were analyzed using the criteria that are listed below. All the results are presented in Tables 4 to 20.

- (i) Best fitness solution: the best of the fitness solution among the solutions obtained during the runs.
- (ii) Mean best fitness solution: this is a measure of the precision (quality) of the result that the algorithm can get within given iterations in all the 25 runs.
- (iii) Standard deviation (Std. Dev.) of mean best fitness solution over 25 runs: this measures the algorithm's stability and robustness.
- (iv) Average number of iterations an algorithm was able to reach the success threshold.
- (v) Success rate (SR) = (Number of successful runs / Total number of runs) $\times$ 100: this is the rate at which the success threshold is met during the independent number of runs and is a reflection of the global search ability and robustness of the algorithm.

Statistical analysis using the Wilcoxon signed rank non-parametric test with 0.05 level of significance [28, 29] was also performed using the numerical results obtained by the algorithms, while box plots were used to analyze their variability in obtaining fitness values in all the runs.

5.4. Results and Discussions. Results obtained from all the experiments are discussed in this subsection to show the overall performance of the various algorithms. Presented in Tables 4, 5, 6, 7, 8, and 9 are the numerical results obtained and used to compare R -PSO_{CLUS} and L -PSO_{CLUS} with GLSPSO. R -PSO_{CLUS} and L -PSO_{CLUS} were also compared with PSOLis using the results presented in Table 10. The results in Tables 11–18 were obtained for the scaled and nonscaled test problems listed in Table 3; the results were used to validate RIW-PSO, R -PSO_{CLUS}, LDIW-PSO, and L -PSO_{CLUS}. In each of the tables, for ease of observation, bold values represent the better results and “-” means that the algorithm could not satisfy the success threshold in any of the runs. The Wilcoxon sign rank nonparametric test, which is used as an alternative to the paired t -test when the results cannot be assumed to be normally distributed, was applied to test the statistical significance differences between RIW-PSO and R -PSO_{CLUS} as well as LDIW-PSO and L -PSO_{CLUS}.

5.4.1. Comparison of R -PSO_{CLUS} and Golden Ratio Local Search Based PSO (GLSPSO). The results in Tables 4–6 show the performance and abilities of R -PSO_{CLUS} and GLSPSO optimizing the test problems over three different problem dimensions. The results of GLSPSO were obtained from [15]. A large problem space was used for all the problems to verify the superiority between the two different local search techniques hybridized with the PSO variants. From the results it is evident that R -PSO_{CLUS} is superior to GLSPSO. Apart from Ackley problem (across the three dimensions) and Rosenbrock (in dimension 100), R -PSO_{CLUS} outperformed GLSPSO. It was able to obtain optimal minimum for some of the problems, demonstrating better exploitation ability, convergence precision, and solution quality.

5.4.2. Comparison between L -PSO_{CLUS} and GLSPSO. To further demonstrate the efficiency of the proposed local search technique, L -PSO_{CLUS} was also implemented and results were compared with the results of GLSPSO obtained from [15]. Three different types of dimensions were also used for the problems. As can be observed in Tables 7–9, across the three dimensions, GLSPSO was only able to perform better than L -PSO_{CLUS} in Ackley problem. Apart from Griewank problem (dimension 10), GLSPSO was outperformed by L -PSO_{CLUS} in the remaining four problems. L -PSO_{CLUS} was able to obtain global optimum for Griewank and Sphere problems across the three dimensions, but, for Rastrigin, it was able to get global minimum for dimension 10. Again, the proposed local search technique demonstrates better exploitation ability than GLSPSO.

5.4.3. Comparison of R -PSO_{CLUS} and L -PSO_{CLUS} with PSOLis. Presented in Table 10 is the result obtained by L -PSO_{CLUS}.

TABLE 3: Benchmark problems.

Number	Problem	Formulation	Feature	Search range
1	Ackley	$f(\vec{x}) = -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e$	MN	± 32
2	Booth	$f(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	MN	± 10
3	Easom	$f(\vec{x}) = -\cos(x_1) \cos(x_2) \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	UN	± 100
4	Griewank	$f(\vec{x}) = \frac{1}{4000} \left(\sum_{i=1}^d x_i^2 \right) - \left(\prod_{i=1}^d \cos \left(\frac{x_i}{\sqrt{i}} \right) \right) + 1$	MN	± 600
5	Dixon-Price	$f(\vec{x}) = (x_1 - 1)^2 + \sum_{i=2}^d i(2x_i^2 - x_{i-1})^2$	UN	± 10
6	Levy	$f(\vec{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{d-1} (y_i - 1)^2 (1 + 10 \sin^2(\pi y_i + 1)) + (y_d - 1)^2 (1 + \sin^2(2\pi x_d))$, where $y_i = 1 + \frac{x_i - 1}{4}$, and $i = 1, 2, \dots, d$	MN	± 10
7	Michalewicz	$f(\vec{x}) = -\sum_{i=1}^d \sin(x_i) \left[\sin \left(\frac{ix_i^2}{\pi} \right) \right]^{2m}$, where $m = 10$	MS	$[0, \pi]$
8	Noisy Quartic	$f(\vec{x}) = \sum_{i=1}^d ix_i^4 + \text{random}(0, 1)$	US	± 1.28
9	Noncontinuous Rastrigin	$f(\vec{x}) = \sum_{i=1}^d (y_i^2 - 10 \cos(2\pi y_i) + 10)$ $y_i = \begin{cases} x_i & \text{if } x_i < 0.5 \\ \frac{\text{round}(2x_i)}{2} & \text{if } x_i \geq 0.5 \end{cases}$	MS	± 5.12
10	Rastrigin	$f(\vec{x}) = \sum_{i=1}^d (x_i^2 - 10 \cos(2\pi x_i) + 10)$	MS	± 5.12
11	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2) + (x_i - 1)^2$	UN	± 30
12	Rotated Ellipsoid	$f(\vec{x}) = \sum_{i=1}^d \left(\sum_{j=1}^i x_j \right)^2$	UN	± 100
13	Salomon	$f(\vec{x}) = -\cos \left(2\pi \sum_{i=1}^d x_i^2 \right) + 0.1 \sqrt{\sum_{i=1}^d x_i^2} + 1$	MN	± 100
14	Schaffer's f6	$f(\vec{x}) = \sum_{i=1}^{d-1} \left(0.5 + \frac{\sin^2(\sqrt{x_{i+1}^2 + x_i^2}) - 0.5}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2} \right)$	MN	± 100
15	Schwefel	$f(\vec{x}) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	MS	± 500
16	Schwefel P2.22	$f(\vec{x}) = \sum_{i=1}^d x_i + \prod_{i=1}^d x_i $	UN	± 10
17	Shubert	$f(\vec{x}) = \prod_{i=1}^d \left(\sum_{j=1}^5 j \cos((j+1) s x_i + j) \right)$	MN	± 10
18	Sphere	$f(\vec{x}) = \sum_{i=1}^d x_i^2$	US	± 100
19	Step	$f(\vec{x}) = \sum_{i=1}^d ([x_i + 0.5])^2$	US	± 10
20	SumSquares		US	± 10
21	Trid	$f(\vec{x}) = \sum_{i=1}^d (x_i - 1)^2 - \sum_{i=2}^d x_i x_{i-1}$	UN	$\pm d^2$

TABLE 4: Comparison between GLSPSO and $R\text{-PSO}_{\text{CLUS}}$ for problems with dimension of 10.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$
Best fitness	0.0364	0.0000	$4.2879e - 04$	$0.0000e + 00$	8.8062	0.0000	2.6188	0.0000	$4.7832e - 04$	$3.1461e - 43$
Mean fitness	0.3413	17.1371	0.0041	0.0016	29.4936	0.0000	9.0025	1.9971	0.0142	0.0000
Worst fitness	1.2653	20.0888	0.0419	0.0791	50.4781	0.0000	18.9887	3.1444	0.0476	0.0000
Std. Dev.	0.2762	6.7543	0.0061	0.0111	10.4372	0.0000	0.034	0.7262	0.0123	0.0000

TABLE 5: Comparison between GLSPSO and $R\text{-PSO}_{\text{CLUS}}$ for problems with dimension of 30.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$
Best fitness	2.2784	20.3075	0.0897	0.0000	109.5946	13.9247	175.8785	22.7589	1.9123	0.0000
Mean fitness	2.8398	20.4778	0.1257	0.0000	185.5221	36.3715	218.4976	27.5147	2.7449	0.0000
Worst fitness	3.2952	20.5792	0.2074	0.0000	229.6229	72.6581	259.2466	76.7433	3.9559	0.0000
Std. Dev.	0.2273	0.0574	0.0274	0.0000	24.9829	16.4882	21.8027	9.9182	0.4840	0.0000

TABLE 6: Comparison between GLSPSO and $R\text{-PSO}_{\text{CLUS}}$ for problems with dimension of 100.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$	GLSPSO	$R\text{-PSO}_{\text{CLUS}}$
Best fitness	3.5148	20.9666	0.3195	0.0022	792.004	293.5795	1378.0	1867.2669	23.0614	0.1970
Mean fitness	3.6709	21.0691	0.4242	0.0230	881.0822	688.0048	1602.0	24909.8486	27.2534	4.7232
Worst fitness	3.7664	21.1306	0.4992	0.0923	934.9773	848.9927	1763.0	95519.4585	29.1615	16.1174
Std. Dev.	0.0551	0.0316	0.0303	0.0255	35.2341	103.1854	90.2874	21083.5791	1.2253	4.2498

TABLE 7: Comparison between GLSPSO and $L\text{-PSO}_{\text{CLUS}}$ for problems with dimension of 10.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$
Best fitness	0.0364	0.0000	$4.2879e - 04$	$0.0000e + 00$	8.8062	0.0000	2.6188	0.0000	$4.7832e - 04$	$6.4151e - 76$
Mean fitness	0.3413	18.2504	0.0041	0.0042	29.4936	0.0000	9.0025	1.0516	0.0142	0.0000
Worst fitness	1.2653	20.0771	0.0419	0.1008	50.4781	0.0000	18.9887	2.8033	0.0476	0.0000
Std. Dev.	0.2762	5.4640	0.0061	0.0186	10.4372	0.0000	0.034	0.6449	0.0123	0.0000

TABLE 8: Comparison between GLSPSO and $L\text{-PSO}_{\text{CLUS}}$ for problems with dimension of 30.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$	GLSPSO	$L\text{-PSO}_{\text{CLUS}}$
Best fitness	2.2784	20.3184	0.0897	0.0000	109.5946	0.1444	175.8785	0.0000	1.9123	0.0000
Mean fitness	2.8398	20.4631	0.1257	0.0000	185.5221	18.7372	218.4976	25.1359	2.7449	0.0000
Worst fitness	3.2952	20.5734	0.2074	0.0000	229.6229	38.8433	259.2466	77.4444	3.9559	0.0000
Std. Dev.	0.2273	0.0615	0.0274	0.0000	24.9829	8.4570	21.8027	13.2536	0.4840	0.0000

TABLE 9: Comparison between GLSPSO and L -PSO_{CLUS} for problems with dimension of 100.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	L -PSO _{CLUS}	GLSPSO	L -PSO _{CLUS}	GLSPSO	L -PSO _{CLUS}	GLSPSO	L -PSO _{CLUS}	GLSPSO	L -PSO _{CLUS}
Best fitness	3.5148	20.2136	0.3195	0.0000	792.004	212.0416	1378.0	93.7390	23.0614	0.0000
Mean fitness	3.6709	21.0491	0.4242	0.0000	881.0822	366.6521	1602.0	107.2300	27.2534	0.0000
Worst fitness	3.7664	21.1152	0.4992	0.0000	934.9773	504.2204	1763.0	428.1758	29.1615	0.0000
Std. Dev.	0.0551	0.1254	0.0303	0.0000	35.2341	68.2009	90.2874	56.9231	1.2253	0.0000

TABLE 10: Comparison between PSOLis, R -PSO_{CLUS} and L -PSO_{CLUS}.

Problem	Algorithm		
	PSOLis	R -PSO _{CLUS}	L -PSO _{CLUS}
Ackley	$4.081e - 03$	$9.263e - 13$	$3.7135e - 15$
Griewank	$2.673e - 02$	$6.921e - 03$	$2.945e - 07$
Rastrigin	2.005	$1.948e - 09$	$8.893e - 06$
Rosenbrock	3.987	$5.180e - 01$	$2.338e - 01$
Sphere	$6.137e - 14$	$2.197e - 27$	$2.401e - 54$

in comparison with result for PSOLis from [18]. Again, the outstanding performance of L -PSO_{CLUS} over its competitor is evidence that the proposed local search technique is very efficient and capable of complementing the global search ability of PSO to obtain quality results by making it overcome premature convergence.

5.4.4. Comparison between RIW-PSO and R -PSO_{CLUS}. The results presented in Table 11 are for the nonscaled test problems as optimized by the two algorithms while those in Tables 12–14 are for the scaled problems with 10, 20, and 30 dimensions, respectively. In Table 19 are the results obtained using the Wilcoxon sign rank nonparametric test.

(1) Results for the Nonscaled Problems. For the 7 nonscaled problems, Table 11 shows that there are no performance differences between the two algorithms in optimizing *Booth*, *Easom*, *Shubert*, and *Trid* problems. For *Michalewicz*, *Schaffer's f6*, and *Salomon*, R -PSO_{CLUS} obtained more quality solutions and demonstrated better global search ability than RIW-PSO. The convergence curves in Figures 1(c) and 1(d) show that R -PSO_{CLUS} has faster and better convergence. However, the P value (0.190) obtained from the Wilcoxon sign test shown in Table 19 revealed that there is no statistical difference in the performance between the two algorithms for the nonscaled problems. Also, the two algorithms have equal median fitness.

(2) Results for 10-Dimensional Problems. For the scaled problems with 10 dimensions, Table 12 clearly reveals great differences in performance between RIW-PSO and R -PSO_{CLUS}. The two algorithms successfully optimized *Rastrigin*, *Rosenbrock*, *Rotated Ellipsoid*, *Schwefel 2.22*, *Sphere*, and *Sum Squares* problems with equal success rate of 100%, but R -PSO_{CLUS} obtained significantly better mean fitness and

standard deviation with fewer number of iterations. R -PSO_{CLUS} was able to obtain the minimum optima for both *Rastrigin* and *Step* problems. For the other problems, R -PSO_{CLUS} clearly outperformed RIW-PSO in solution quality, convergence precision, global search ability, and robustness, though none of them could meet the success threshold in optimizing the *Noisy Quartic* problem. The P value (0.001) obtained from the Wilcoxon sign test presented in Table 19 indicates that there is statistically significant difference in performance between the two algorithms with a large effect size of $r = 0.6$ in favour of R -PSO_{CLUS}. The median fitness is also an evidence of this.

(3) Results for 20-Dimensional Problems. The same set of experiments was performed using the same scaled problems but with their dimensions increased to 20, which also increased their complexities except *Griewank*. The numerical results in Table 13 also show that there are great differences in performance between RIW-PSO and R -PSO_{CLUS}. The two algorithms had equal success rate of 100% in optimizing *Rosenbrock*, *Schwefel 2.22*, *Sphere*, and *Sum Squares* problems with R -PSO_{CLUS} obtaining significantly better mean fitness (except *Rosenbrock*), standard deviation, and fewer number of iterations. Out of the remaining 10 problems R -PSO_{CLUS} outperformed RIW-PSO in 9 of them with better solution quality, convergence precision, global search ability, and robustness; it also had success rate of 100% in 6 of the problems compared with RIW-PSO and was able to obtain global minimum for *Griewank* and *Step* problems. The algorithms could not meet the success criteria in optimizing the *Dixon-Price*, *Noisy Quartic*, and *Schwefel* problems, but R -PSO_{CLUS} still performed better than RIW-PSO. The P value (0.023) from Wilcoxon sign test as shown in Table 19 also indicates that there is statistically significant difference in performance between the two algorithms with a medium effect size of $r = 0.43$ in favour of R -PSO_{CLUS}. The median fitness value of R -PSO_{CLUS} is smaller than that of RIW-PSO.

(4) Results for 30-Dimensional Problems. Table 14 represents the experimental results obtained by the two algorithms using the same scaled problems but with their dimensions scaled to 30, which further increased their complexities except *Griewank*. The results further show the great differences in performance between RIW-PSO and R -PSO_{CLUS}. Out of the 14 problems R -PSO_{CLUS} had 100% success rate in 7 of them (4 multimodal and 3 unimodal) while RIW-PSO could only have in 3 of them (all unimodal). The two algorithms

TABLE II: Results for RIW-PSO and R-PSO_{CLUS} for the 7 nonscaled benchmark problems.

Problem	Booth		Easom		Michalewicz		Schaffer's f6		Salomon		Shubert		Trid-6	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	0.0000e + 00	0.0000e + 00	-1.0000e + 00	-1.0000e + 00	-3.3453e + 00	-4.4371e + 00	0.0000e + 00	0.0000e + 00	9.9833e + 02	7.3608e + 02	-1.8673e + 02	-1.8673e + 02	-5.0000e + 00	-5.0000e + 00
Mean fitness	0.0000e + 00	0.0000e + 00	-1.0000e + 00	-1.0000e + 00	-2.6034e + 00	-4.1008e + 00	4.7052e + 03	1.1659e + 03	9.9833e + 02	7.0073e + 02	-1.8673e + 02	-1.8673e + 02	-5.0000e + 00	-5.0000e + 00
Std. Dev.	0.0000e + 00	0.0000e + 00	6.6613e + 00	6.6613e + 00	4.2719e + 01	1.7866e + 01	4.8183e + 03	3.1573e + 03	6.2063e + 18	4.4236e + 02	3.7706e + 14	4.0594e + 14	8.6628e + 14	7.0555e + 14
Av. iteration	39.12	37.92	55.0	45.48	—	—	133.83	109.95	—	923.40	71.8	107.16	114.40	110.20
SR (%)	100	100	100	100	0	0	48	88	0	20	100	100	100	100

TABLE 12: Results for RIW-PSO and R-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 10.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	3.9968e-15	4.4409e-16	7.3960e-03	0.0000e+00	2.4652e-31	2.4652e-31	1.4997e-32	1.4997e-32	5.9308e-04	2.7756e-05	6.0002e+00	0.0000e+00	3.9767e+00	0.0000e+00
Mean fitness	1.0316e-01	2.8599e-02	6.6238e-02	2.8539e-03	6.1333e-01	6.1333e-01	1.4997e-01	1.4997e-01	4.6455e-03	1.1590e-04	1.1640e+01	7.1054e-17	1.2208e+01	0.0000e+00
Std. Dev.	5.0537e-01	1.6573e-15	3.3218e-02	1.3981e-02	1.8086e-01	1.8086e-01	7.5307e-01	0.0000e+00	3.2781e-03	5.9530e-05	4.3533e+00	3.4809e-16	4.4684e+00	0.0000e+00
Av. iteration	287.88	263.68	—	464.16	295.50	258.50	127.31	99.32	—	—	40.49	12.44	49.44	25.00
SR (%)	96	100	0	96	8	8	52	100	0	0	92	100	100	100
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	4.6541e-03	3.7243e-01	2.5690e-27	5.9210e-29	-3.2818e+03	-4.1898e+03	1.8348e-32	2.7555e-34	7.3673e-61	5.4496e-61	0.0000e+00	0.0000e+00	8.1835e-62	1.9089e-61
Mean fitness	1.4459e+00	7.1832e-01	1.0457e-21	3.5437e-24	-2.6199e+03	-4.1384e+03	3.3382e-28	1.2830e-30	4.1760e-53	9.2787e-55	8.0000e+01	0.0000e+00	2.1909e+53	9.3329e-54
Std. Dev.	1.6362e+00	2.3127e-01	3.0984e-21	1.1617e-23	3.3350e+02	9.7987e+01	1.3406e-27	4.6701e-30	1.2467e-52	3.6764e-54	1.9183e+00	0.0000e+00	1.0052e+52	3.3839e-53
Av. iteration	89.28	35.16	426.60	337.16	—	877.69	268.84	222.04	180.12	158.08	88.72	15.68	143.52	123.48
SR (%)	100	100	100	100	0	52	100	100	100	100	72	100	100	100

TABLE 13: Results for RIW-PSO and R-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 20.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	7.5495e-15	3.9968e-15	0.0000e+00	0.0000e+00	6.6667e-01	6.6667e-01	1.4997e-32	1.4997e-32	3.9758e-03	8.2784e-05	1.2000e+01	1.6698e-03	1.4000e+01	9.8524e-03
Mean fitness	2.6017e-01	3.9968e-01	2.7935e-02	0.0000e+00	6.6667e-01	6.6667e-01	7.8421e-01	1.4997e-32	1.0286e-02	2.7771e-04	3.0521e+01	4.4448e+00	2.8521e+01	4.3599e+00
Std. Dev.	6.0386e-01	0.0000e+00	1.8216e-02	0.0000e+00	2.4222e-16	4.1352e-08	8.0230e-01	0.0000e+00	5.0796e-03	1.3566e-04	1.0922e+01	2.5668e+00	9.5087e+00	1.9684e+00
Av. iteration	580.67	488.52	402.00	382.80	—	—	248.00	200.04	—	—	51.5	86.72	56.2	89.08
SR (%)	84	100	4	100	0	0	16	100	0	0	16	100	20	100
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	8.2827e-01	7.4672e+00	8.7694e-04	7.9922e-09	-5.9318e+03	-8.1903e+03	2.4775e-22	3.5223e-26	3.1840e-43	2.9531e-44	0.0000e+00	0.0000e+00	4.8722e+45	2.1277e-46
Mean fitness	9.1608e+00	1.0488e+01	7.2146e-06	1.9591e-05	-4.5638e+03	-7.0464e+03	8.5083e-17	4.9420e-23	3.9759e-37	8.4151e-38	1.5600e+00	0.0000e+00	8.3393e+39	1.4859e-39
Std. Dev.	3.5710e+00	8.1620e-01	1.9200e-05	7.2466e-05	6.0867e+02	8.4004e+02	3.9686e-16	1.2554e-22	1.2218e-36	3.0436e-37	1.6020e+00	0.0000e+00	1.4280e+38	4.9918e-39
Av. iteration	263.40	133.72	1763.10	1636.74	—	—	575.92	418.88	354.00	309.04	646.57	19.36	310.72	268.12
SR (%)	100	100	84	92	0	0	100	100	100	100	28	100	100	100

TABLE 14: Results for RIW-PSO and R-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 30.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	1.4655e – 14	3.9968e – 15	0.0000e + 00	0.0000e + 00	6.6667e – 01	6.6667e – 01	2.6858e – 01	1.4997e – 32	3.3454e – 03	1.0873e – 04	1.9001e + 01	5.7746e – 02	1.7895e + 01	0.0000e + 00
Mean fitness	5.3345e – 01	4.1389e – 02	1.1200e – 02	0.0000e + 00	6.6667e – 01	6.6716e – 01	1.8286e + 00	1.4997e – 32	1.1252e – 02	3.1438e – 04	4.3331e + 01	9.0806e + 00	3.2489e + 01	6.5810e + 00
Std. Dev.	8.1543e – 01	6.9619e – 16	1.4757e – 02	0.0000e + 00	2.4146e – 03	2.4146e – 03	1.1781e + 00	0.0000e + 00	4.3926e – 03	1.1754e – 04	1.8889e + 01	5.1000e + 00	1.0662e + 01	5.7209e + 00
Av. iteration	959.88	776.20	628.00	525.48	—	—	—	348.76	—	—	107	377.72	141.00	379.35
SR (%)	64	100	36	100	0	0	0	100	0	0	4	100	4	92
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Algorithm	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}	RIW-PSO	R-PSO _{CLUS}
Best fitness	4.0555e – 02	1.8942e + 01	3.2599e – 04	1.7148e – 04	–7.9654e + 03	–1.0692e + 04	2.7819e – 17	1.0011e – 24	5.7947e – 39	5.0490e – 42	0.0000e + 00	0.0000e + 00	1.7310e – 36	1.2387e – 44
Mean fitness	2.3794e + 01	1.9930e + 01	5.5035e – 03	1.8327e – 02	–6.7376e + 03	–9.1033e + 03	4.1597e – 13	6.4368e – 23	4.1277e – 33	3.6673e – 36	3.7600e + 00	0.0000e + 00	3.9240e – 33	5.9219e – 39
Std. Dev.	1.9052e + 01	5.2276e – 01	8.1952e – 03	4.4826e – 02	8.7747e + 02	8.6528e + 02	1.5547e – 12	7.6462e – 23	1.1473e – 32	8.2055e – 36	2.3200e + 00	0.0000e + 00	1.3909e – 32	1.2917e – 38
Av. iteration	1523.36	2813.14	—	—	—	—	1139.64	658.68	595.52	505.80	2881.00	21.48	526.28	436.08
SR (%)	44	56	0	0	0	0	100	100	100	100	4	100	100	100

TABLE 15: Results for LDIW-PSO and L-PSO_{CLUS} for the 7 nonscaled benchmark problems.

Problem	Booth		Easom		Michalewicz		Schaffer's f6		Salomon		Shubert		Trid-6	
	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	0.0000e + 00	0.0000e + 00	-1.0000e + 00	-1.0000e + 00	-3.4792e + 00	-4.3762e + 00	0.0000e + 00	0.0000e + 00	9.9833e - 02	3.3189e - 65	-1.8673e + 02	-1.8673e + 02	-5.0000e + 01	-5.0000e + 01
Mean fitness	0.0000e + 00	0.0000e + 00	-1.0000e + 00	-1.0000e + 00	-2.6736e + 00	-4.1056e + 00	1.5545e - 03	1.1659e - 03	9.9833e - 02	8.7853e - 02	-1.8673e + 02	-1.8673e + 02	-50000e + 01	-50000e + 01
Std. Dev.	0.0000e + 00	0.0000e + 00	6.6613e - 16	6.6613e - 16	3.6921e - 01	1.6196e - 01	3.5619e - 03	3.1573e - 03	7.3434e - 18	3.2442e - 02	3.9382e - 14	7.5559e - 05	7.8572e - 14	9.3154e - 14
Av. iteration	79.48	81.16	107.16	73.12	—	—	279.90	174.14	—	742	175.52	208.20	287.48	281.72
SR (%)	100	100	100	100	0	0	84	88	0	12	100	96	100	100

TABLE 16: Results for LDIW-PSO and L-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 10.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	3.9968e-15	4.4409e-16	2.9510e-02	0.0000e+00	6.6667e-01	6.6667e-01	1.4997e-31	1.4997e-32	3.4641-04	1.1331e-05	4.0001e+00	0.0000e+00	2.9825e+00	0.0000e+00
Mean fitness	4.9916e-15	2.4335e-15	8.4720e-02	1.4639e-06	6.6667e-01	6.6667e-01	1.9892e-01	1.4997e-32	2.9606e-03	9.7778e-05	1.2200e+01	4.0007e-02	9.9019e+00	0.0000e+00
Std. Dev.	1.5952e-15	1.7635e-15	2.7646e-02	7.1705e-06	1.5401e-10	1.2278e-12	4.4094e-01	0.0000e+00	2.0277e-03	6.3873e-05	6.2484e+00	1.9600e-01	5.2190e+00	0.0000e+00
Av. iteration	500.92	486.84	—	703.83	—	—	309.58	281.36	—	—	40.86	12.88	45.13	26.24
SR (%)	100	100	0	96	0	0	76	100	0	0	88	100	96	100
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	2.7701e-03	1.5581e-01	2.1928e-44	1.2232e-52	-3.2028e+03	-4.1898e+03	3.3962e-48	2.3988e-46	9.8201e-110	3.2307e-112	0.0000e+00	0.0000e+00	2.3361e-110	1.6446e-112
Mean fitness	9.3256e-01	7.2655e-01	5.8802e-32	4.6946e-47	-2.5079e+03	-4.1317e+03	8.0458e-23	5.4599e-29	3.1249e-101	1.7383e-106	1.6000e-01	0.0000e+00	1.6238e-105	6.7305e-105
Std. Dev.	1.3984e+00	5.1516e-01	2.8672e-31	1.6736e-46	4.0568e+02	1.2489e+02	3.9336e-22	2.6683e-28	1.2642e-100	3.3735e-106	7.8384e-01	0.0000e+00	5.1760e-105	3.1425e-104
Av. iteration	155.20	69.64	566.44	559.20	—	798.45	478.12	450.76	385.52	374.28	73.79	15.76	340.56	327.24
SR (%)	100	100	100	100	0	44	100	100	100	100	96	100	100	100

TABLE 17: Results for LDIW-PSO and L-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 20.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	1.4655e-14	3.9968e-15	7.3960e-03	0.0000e+00	6.6667e-01	6.6667e-01	1.4997e-32	1.4997e-32	8.8724e-04	8.2862e-05	9.0003e-00	5.4958e-02	7.8533e-00	0.0000e+00
Mean fitness	2.7869e-13	3.9968e-15	3.5766e-02	1.1969e-05	6.6667e-01	6.6667e-01	9.4811e-01	1.4997e-32	6.2935e-03	1.8753e-04	2.9441e-01	5.0733e+00	2.2627e-01	1.5954e+00
Std. Dev.	1.1016e-12	0.0000e+00	2.3388e-02	3.7730e-05	9.4022e-16	2.6746e-08	1.0290e-00	0.0000e+00	2.8130e-03	9.1506e-05	1.1371e-01	2.4613e+00	7.0594e-00	2.9944e+00
Av. iteration	785.08	773.48	—	813.50	—	—	545.13	518.48	—	—	93	88.40	256.17	144.32
SR (%)	100	100	0	88	0	0	32	100	0	0	16	100	48	100
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	8.5273e-01	7.8279e+00	5.7451e-10	3.1410e-20	-5.5568e+03	-79200e+03	2.1114e-17	4.5537e-28	3.4892e-59	4.4204e-68	0.0000e+00	0.0000e+00	2.7411e-60	3.2424e-71
Mean fitness	8.8496e+00	9.1318e+00	6.1269e-05	1.7093e-15	-4.5047e+03	-68180e+03	3.5766e-09	1.6526e-23	1.0393e-45	1.6274e-58	8.0000e-02	0.0000e+00	1.4838e-48	1.5378e-59
Std. Dev.	3.4651e-00	9.4307e-01	1.7066e-04	4.8692e-15	5.1886e+02	5.0603+02	8.6693e-09	3.1511e-23	3.1193e-45	5.5196e-58	2.7129e-01	0.0000e+00	5.6876e-48	7.4256e-59
Av. iteration	594.96	419.12	1495.45	1254.52	—	—	820.56	722.76	639.88	636.08	147.52	19.88	595.80	593.52
SR (%)	100	100	88	100	0	0	100	100	100	100	92	100	100	100

TABLE 18: Results for LDIW-PSO and L-PSO_{CLUS} for the 14 scaled benchmark problems with dimension of 30.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	3.6993e-13	3.9968e-15	2.2204e-16	0.0000e+00	6.6667e-01	6.6667e-01	3.4554e-28	1.4997e-32	3.2281e-03	8.0745e-05	2.0001e-01	2.2356e+00	1.8889e-01	0.0000e+00
Mean fitness	5.6071e-11	3.9968e-15	1.2289e-02	0.0000e+00	6.6667e-01	6.6667e-01	2.0125e+00	1.4997e-32	7.2161e-03	2.3200e-04	4.1361e-01	1.2883e+01	3.3921e-01	8.3217e+00
Std. Dev.	2.1743e-10	0.0000e+00	1.5738e-02	0.0000e+00	3.8233e-12	3.1157e-14	1.6464e+00	0.0000e+00	3.4677e-03	1.0514e-04	1.3494e+01	5.1402e+00	9.7583e-00	7.6697e+00
Av. iteration	1245.68	1194.40	1050.23	1060.44	—	—	928.00	865.44	—	—	—	658.48	518.00	718.21
SR (%)	100	100	52	100	0	0	4	100	0	0	0	92	4	96
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}	LDIW-PSO	L-PSO _{CLUS}
Best fitness	1.4208e-01	1.4715e+01	1.4959e-04	3.8064e-11	-7.7298e-03	-1.0232e+04	6.6416e-10	1.2902e-24	9.3155e-41	1.4592e-57	0.0000e+00	0.0000e+00	1.0873e-41	1.2031e-57
Mean fitness	1.8350e+01	1.7789e+01	4.5745e-02	1.8073e-06	-6.5026e-03	-8.6435e+03	1.0569e-04	2.4455e-22	1.2325e-30	6.2108e-47	7.2000e-01	0.0000e+00	1.6409e-28	2.0496e-48
Std. Dev.	1.1240e+01	1.8867e+00	1.3691e-01	4.7189e-06	7.1966e-02	1.0203e+03	4.0792e-04	2.8267e-22	3.7363e-30	3.0419e-46	8.7270e-01	0.0000e+00	5.9071e-28	9.3904e-48
Av. iteration	1920.22	2078.70	—	2524.35	—	—	1576.75	1120.20	1033.04	1028.12	408.92	22.52	968.72	952.28
SR (%)	92	92	0	92	0	0	80	100	100	100	52	100	100	100

TABLE 19: Wilcoxon signed rank test on mean fitness obtained by RIW-PSO and $R\text{-PSO}_{\text{CLUS}}$ for the test problems.

Measurement	Scaled problems			Nonscaled problems
	Dim = 10	Dim = 20	Dim = 30	
$R\text{-PSO}_{\text{CLUS}} < \text{RIW-PSO}$	13	11	12	3
$R\text{-PSO}_{\text{CLUS}} > \text{RIW-PSO}$	0	2	2	0
$R\text{-PSO}_{\text{CLUS}} = \text{RIW-PSO}$	1	1	0	4
z	-3.190	-2.274	-2.606	-1.604
P value	0.001	0.023	0.009	0.190
r	0.600	0.430	0.490	—
Median				
RIW-PSO	0.847	0.144	0.272	-1.000
$R\text{-PSO}_{\text{CLUS}}$	0.000	0.000	0.000	-1.000

TABLE 20: Wilcoxon signed rank test on mean fitness obtained by LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$ for the test problems.

Measurement	Scaled problems			Nonscaled
	Dim = 10	Dim = 20	Dim = 30	
$L\text{-PSO}_{\text{CLUS}} < \text{LDIW-PSO}$	12	12	13	3
$L\text{-PSO}_{\text{CLUS}} > \text{LDIW-PSO}$	1	1	0	0
$L\text{-PSO}_{\text{CLUS}} = \text{LDIW-PSO}$	1	1	1	4
z	-2.988	-2.552	-3.181	-1.604
P value	0.003	0.011	0.001	0.190
r	0.565	0.482	0.601	—
Median				
LDIW-PSO	0.044	0.021	0.029	-1.000
$L\text{-PSO}_{\text{CLUS}}$	0.000	0.000	0.000	-1.000

had equal success rate of 100% in optimizing *Schwefel 2.22*, *Sphere*, and *Sum Squares* problems with $R\text{-PSO}_{\text{CLUS}}$ obtaining significantly better mean fitness standard deviation and fewer number of iterations. Optimizing *Dixon-Price*, *Noisy Quartic*, *Rotated Ellipsoid*, and *Schwefel* problems, none of the algorithms could meet the success criteria, yet $R\text{-PSO}_{\text{CLUS}}$ still obtained better results than RIW-PSO. In all the 14 problems except *Rotated Ellipsoid*, $R\text{-PSO}_{\text{CLUS}}$ outperformed RIW-PSO and was able to obtain global minimum for *Griewank* and *Step* problems. The P value (0.009) from Wilcoxon sign test in Table 19 is a confirmatory evidence that there is statistically significant difference in performance between RIW-PSO and $R\text{-PSO}_{\text{CLUS}}$ with a large effect size of $r = 0.49$ in favour of $R\text{-PSO}_{\text{CLUS}}$. The median fitness value of $R\text{-PSO}_{\text{CLUS}}$ is also smaller than that of RIW-PSO. The convergence graphs of six 30-dimensional test problems shown in Figure 2 demonstrate the speed and ability of convergence of the two algorithms. From the graphs it is clear that $R\text{-PSO}_{\text{CLUS}}$ demonstrates better convergence and global search ability than RIW-PSO. Besides it also possesses better ability to get out of local optima.

5.4.5. Comparison between LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$. Presented in Table 15 are the results for the nonscaled test problems as optimized by the two algorithms while those in Tables 16–18 are for the scaled problems with 10, 20, and 30 dimensions, respectively. The statistical analysis done by

applying Wilcoxon sign rank nonparametric test is presented in Table 20.

(1) *Results for the Nonscaled Problems.* Results in Table 15 show that there are no clear performance differences between LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$ in optimizing *Booth*, *Easom*, *Shubert*, and *Trid* problems; however, there are some not too significant differences in their average number of iterations to reach the success thresholds and standard deviation; in *Shubert*, LDIW-PSO obtained 100% success but $L\text{-PSO}_{\text{CLUS}}$ could not. Figures 1(a), 1(b), 1(e), and 1(f) show their convergence behaviour. Optimizing *Michalewicz*, *Schaffer's f6*, and *Salomon*, $L\text{-PSO}_{\text{CLUS}}$ obtained better quality solutions and has better search ability than LDIW-PSO. Also, the convergence graphs in Figures 1(c) and 1(d) show that $L\text{-PSO}_{\text{CLUS}}$ have faster and better convergence in *Schaffer's f6* and *Salomon*. The curves show that the two algorithms were trapped in local optima as shown by the flat parts of their curves and were able to escape from some of them. The P value (0.190) in Table 20, obtained from the Wilcoxon sign test, indicates that there is no statistically significant difference between the two algorithms in performance.

(2) *Results for 10-Dimensional Problems.* Optimizing the 10-dimensional scaled problems, $L\text{-PSO}_{\text{CLUS}}$ had 100% success in 10 of the 14 problems (4 multimodal and 6 unimodal) while LDIW-PSO had 100% success in 6 problems (1 multimodal and 5 unimodal) as shown in Table 16. It is only $L\text{-PSO}_{\text{CLUS}}$

FIGURE 1: Convergence graphs for 6 of the nonscaled benchmark problems.

FIGURE 2: Convergence graphs for 6 of the scaled benchmark problems with dimension of 30.

FIGURE 3: Box plots for 6 of the scaled test problems.

that could successfully obtain the minimum optima for both *Rastrigin* and *Step* problems but none could reach the success threshold for *Dixon-Price* and *Noisy Quartic*. In all the problems except *Dixon-Price* (where they have approximately equal performance) and *Sum Squares*, $L\text{-PSO}_{\text{CLUS}}$

clearly outperformed LDIW-PSO in obtaining better solution quality, convergence precision, global search ability, and robustness as well as fewer number of iterations. To confirm this, Wilcoxon sign test was performed on the mean best fitness over all the problems and results are presented in

Table 20; the P value (0.003) obtained indicates that there is statistically significant difference in performance between the two algorithms with a large effect size of $r = 0.565$ in favour of $L\text{-PSO}_{\text{CLUS}}$ which also has a lower median value for the mean best fitness.

(3) *Results for 20-Dimensional Problems.* The numerical results in Table 17 also show that there are great differences in performance between LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$ performing the same set of experiments but with the problems dimensions increased to 20. The two algorithms had equal success rate of 100% in optimizing *Ackley*, *Rosenbrock*, *Schwefel 2.22*, *Sphere*, and *Sum Squares* problems with $L\text{-PSO}_{\text{CLUS}}$ obtaining significantly better mean fitness (except *Rosenbrock*), standard deviation, and fewer number of iterations. $L\text{-PSO}_{\text{CLUS}}$ outperformed LDIW-PSO in 7 (5 multimodal and 2 unimodal) of the rest 9 problems and obtained better solution, convergence precision, global search ability, and robustness; it was also able to obtain global minimum for *Step* problem. The algorithms could not reach the success thresholds for *Dixon-Price*, *Noisy Quartic*, and *Schwefel* problems. The nonparametric test that was performed using Wilcoxon sign test, with results shown in Table 20, also confirms statistically significant difference in performance between the two algorithms with P value (0.011) and a large effect size of $r = 0.482$ in the direction of $L\text{-PSO}_{\text{CLUS}}$. The median fitness value of $L\text{-PSO}_{\text{CLUS}}$ is also smaller than that of LDIW-PSO.

(4) *Results for 30-Dimensional Problems.* Scaling the dimensions of test problems to 30 to further increase their complexities, except *Griewank* which decreases in complexity with increased dimension, did not affect the better performance of $L\text{-PSO}_{\text{CLUS}}$ over LDIW-PSO. Presented in Table 18 are the experimental results obtained by the two algorithms optimizing the same scaled problems. The results indicate that there are great differences between LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$ in performance. Out of the 14 problems $L\text{-PSO}_{\text{CLUS}}$ had 100% success rate in 6 of them (3 multimodal and 3 unimodal) while LDIW-PSO could only have in 3 (1 multimodal and 2 unimodal). They had equal success rate of 100% in optimizing *Ackley*, *Sphere*, and *Sum Squares* problems and 92% in *Rosenbrock* with $L\text{-PSO}_{\text{CLUS}}$ obtaining significantly better results. Optimizing *Dixon-Price*, *Noisy Quartic*, and *Schwefel* problems, none of the algorithms could reach the success threshold, yet $L\text{-PSO}_{\text{CLUS}}$ still obtained better results than LDIW-PSO, except in *Dixon-Price* where they had approximately the same performance. LDIW-PSO was not able to reach success threshold for *Noncontinuous Rastrigin* and *Rotated Ellipsoid* problems unlike $L\text{-PSO}_{\text{CLUS}}$. In all the 14 problems $L\text{-PSO}_{\text{CLUS}}$ conceded in none to LDIW-PSO and it was able to obtain global minimum for *Griewank* and *Step* problems. The P value (0.001) in Table 20 further confirms that there is statistically significant difference between LDIW-PSO and $L\text{-PSO}_{\text{CLUS}}$ with a large effect size of $r = 0.601$ in the direction of $L\text{-PSO}_{\text{CLUS}}$. The median value for the mean fitness of $L\text{-PSO}_{\text{CLUS}}$ is also smaller than that of RIW-PSO. Figure 1 shows the convergence graphs of the two algorithms.

From the graphs it is clear that $L\text{-PSO}_{\text{CLUS}}$ demonstrates better convergence speed, better ability to escape premature convergence, and global search ability than LDIW-PSO.

5.4.6. *Box Plots Analysis.* Other than using statistical test to observe the performance of RIW-PSO, $R\text{-PSO}_{\text{CLUS}}$, LDIW-PSO, and $L\text{-PSO}_{\text{CLUS}}$, box plots analysis was also performed for 6 of the scaled test problems with 30 dimensions; the results are presented in Figures 3(a)–3(f). Box plots give a direct visual comparison of both location and the dispersion of data. The four algorithms are plotted together to optimize space. In each of the plot, RIW-PSO is compared with $R\text{-PSO}_{\text{CLUS}}$ while LDIW-PSO is compared with $L\text{-PSO}_{\text{CLUS}}$. The plots strengthen and justify the better performance of PSO when used with the proposed local search technique.

6. Conclusion

A new local search technique has been proposed in this paper with the goal of addressing the problem of premature convergence associated with particle swarm optimization algorithms. The proposed local search was used to efficiently improve the performance of two existing PSO variants, RIW-PSO and LDIW-PSO. These variants have been known to be less efficient optimizing continuous optimization problems. In this paper they were hybridized with the local search to form two other variants $R\text{-PSO}_{\text{CLUS}}$ and $L\text{-PSO}_{\text{CLUS}}$. Some well-studied benchmark problems with low and high dimensions were used to extensively validate the performance of these new variants and comparisons were made with RIW-PSO and LDIW-PSO. They were also compared with two other PSO variants in the literature, which are hybridized with different local search techniques. The experimental results obtained show that the proposed variants successfully obtain better results with high quality while demonstrating better convergence velocity and precision, stability, robustness, and global-local search ability than the competing variants. This therefore shows that the local search technique proposed can help PSO algorithms execute effective exploitation in the search space to obtain high quality results for complex continuous optimization problems. This local search technique can be used with any population-based optimization algorithms to obtain quality solutions to simple and complex optimization problem.

Further study is needed on the parameter tuning of the proposed local search technique. Empirical investigation of the behaviour of the technique in optimizing problems with noise needs further study. The scalability of the algorithms for problems with higher dimension greater than 100 is essential. Finally, the proposed algorithm can be applied to real-world optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors acknowledge College of Agriculture, Engineering and Sciences, University of Kwazulu-Natal, South Africa, for their support towards this work.

References

- [1] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.
- [2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.
- [3] V. N. Dieu, P. Schegner, and W. Ongsakul, "A newly improved particle swarm optimization for economic dispatch with valve point loading effects," in *Proceedings of the IEEE Power and Energy Society General Meeting*, pp. 1–8, July 2011.
- [4] J. Bai, X. Zhang, and Y. Guo, "Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA '09)*, pp. 4754–4759, August 2009.
- [5] M. M. Mansour, S. F. Mekhamer, and N. E.-S. El-Kharbawe, "A modified particle swarm optimizer for the coordination of directional overcurrent relays," *IEEE Transactions on Power Delivery*, vol. 22, no. 3, pp. 1400–1410, 2007.
- [6] A. M. Arasomwan and A. O. Adewumi, "An adaptive velocity particle swarm optimization for high-dimensional function optimization," in *Proceedings of the IEEE Congress Evolutionary Computation (CEC '13)*, pp. 2352–2359, 2013.
- [7] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.
- [8] G. I. Evers, *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization [M.S. thesis]*, Graduate School of the University of Texas-Pan American, 2009.
- [9] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, vol. 1447, pp. 591–600, San Diego, Calif, USA, March 1998.
- [10] R. Akbari and K. Ziarati, "Combination of particle swarm optimization and stochastic local search for multimodal function optimization," in *Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08)*, vol. 2, pp. 388–392, December 2008.
- [11] C. Junying, Q. Zheng, L. Yu, and L. Jiang, "Particle swarm optimization with local search," in *Proceedings of the International Conference on Neural Networks and Brain (ICNNB '05)*, pp. 481–484, October 2005.
- [12] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons and Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.
- [13] A. A. Mousa, M. A. El-Shorbagy, and W. F. Abd-El-Wahed, "Local search based hybrid particle swarm optimization algorithm for multiobjective optimization," *Swarm and Evolutionary Computation*, vol. 3, pp. 1–14, 2012.
- [14] H. Pan, X. Han, and M. Zheng, "Particle swarm-simulated annealing fusion algorithm and its application in function optimization," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, vol. 21, pp. 78–81, December 2008.
- [15] Y. Sun, B. J. Wyk, and Z. Wang, "A new golden ratio local search based particle swarm optimization," in *Proceedings of the International Conference on Systems and Informatics (ICSAI '12)*, pp. 754–757, 2012.
- [16] J. Tang and X. Zhao, "Particle swarm optimization using adaptive local search," in *Proceedings of the International Conference on Future BioMedical Information Engineering (FBIE '09)*, pp. 300–303, December 2009.
- [17] Y.-J. Wang, "Improving particle swarm optimization performance with local search for high-dimensional function optimization," *Optimization Methods and Software*, vol. 25, no. 5, pp. 781–795, 2010.
- [18] W.-B. Zhang, J.-Y. Chen, and Y.-Q. Ye, "Study on particle swarm optimization algorithm with local interpolation search," in *Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR '10)*, vol. 1, pp. 345–348, March 2010.
- [19] A. M. Arasomwan and A. O. Adewumi, "On the performance of linear decreasing inertia weight particle swarm optimization for global optimization," *The Science World Journal*, vol. 2013, Article ID 860289, 2013.
- [20] J. Ememipour, M. M. S. Nejad, M. M. Ebadzadeh, and J. Rezanejad, "Introduce a new inertia weight for particle swarm optimization," in *Proceedings of the IEEE 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT '09)*, pp. 1650–1653, November 2009.
- [21] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, September 2007.
- [22] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3658–3670, 2011.
- [23] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the of the Congress on Evolutionary Computation*, vol. 1, pp. 94–100, Seoul, Korea, May 2001.
- [24] M. E. H. Pedersen, *Tuning & simplifying heuristical optimization [Ph.D. thesis]*, School of Engineering Sciences, University of Southampton, Southampton, UK, 2010.
- [25] S. Chetty and A. O. Adewumi, "Three new stochastic local search algorithms for continuous optimization problems," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 675–721, 2013.
- [26] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.
- [27] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained global optimization," *Optimization Methods and Software*, vol. 26, no. 6, pp. 945–970, 2011.
- [28] C. Dytham, *Choosing and Using Statistics: A Biologist's Guide*, Wiley-blackwell, Malaysia, 3rd edition, 2011.
- [29] J. Pallant, *SPSS Survival Manual*, McGraw-Hill, Singapore, 4th edition, 2010.