Hierarchical Particle Swarm Optimization for Optimization Problems

Chia-Chong Chen

Department of Electronics Engineering, Wufeng Institute of Technology, Chiayi, Taiwan, R.O.C.

Abstract

In this paper, a hierarchical particle swarm optimization (HPSO) is proposed to improve the premature convergence in the PSO approach. In the proposed HPSO approach, all particles are arranged in a regular tree structure and move up or down in the tree based on their fitness value. For the velocity update of each particle, it depends on the position of each particle in the tree. Besides, a mutation operator is added into the proposed HPSO approach. Consequently, the diversity of the population increases so that the HPSO approach can improve the premature convergence in the PSO approach. Finally, several benchmark functions for optimization problems are utilized to illustrate the effectiveness of the proposed HPSO approach to improving the premature convergence.

Key Words: Evolutionary Algorithm, Hierarchical Particle Swarm Optimization, Optimization Problem

1. Introduction

During the last several decades, there has been a growing interest in evolutionary computing, which has inspired from the mechanisms of natural evolution. According to the common idea of all these evolutionary algorithms, the environmental pressure causes natural selection and this makes the fitness of population rise. Therefore, the evolutionary algorithm is an effective tool to solve optimization problems. Among existing evolutionary algorithms, genetic algorithm (GA) [1] and particle swarm optimization (PSO) [2,3] are the well-known tools for solving optimization problems. Genetic algorithm is a method to obtain an optimal solution by applying a theory inspired by biological evolution. It employs the Darwinian survival-of-the-fittest theory to yield the best of the characters among the population and performs a random information exchange to produce better individuals. The GA uses reproduction, crossover and mutation operators to search the global optimum solution in the solution space. In the search of the GA, the mutation operator works for exploration and the reproduction and crossover operators work for exploitation. Therefore, the GA approach provides a way to possibly obtain the global optimum solution. However, when the number of the parameters for the considered optimization problem is very large, the solution space will increase enormously so that the expense of the evolutionary computation will become almost impractical. Moreover, the GA is not effective for searching the solution space locally due to crossover-based-search, and the diversity of the population sometimes decreases rapidly. These will lead to the lack of the local search ability and the premature convergence. In order to improve the local search ability, GA with the one-point crossover operator (OEGA), GA with the two-point crossover operator (TEGA), GA with the uniform crossover operator (UEGA), GA with the BLX-α crossover operator (BLXGA) [4] and GA with the orthogonal crossover operator [5] are proposed. In 1995,

*Corresponding author. E-mail: ccchen@mail.wfc.edu.tw

the PSO idea was originally introduced by Kennedy and Eberhart as an evolutionary computation technique inspired by swarm intelligence such as bird flocking, fish schooling and even human social behavior. The PSO is initialized with a population of random solutions of the fitness function. The individuals in the population are called as particles. The position of each particle is updated by a new velocity which is based on its previous velocity, its personal best position and the global best position. According to the velocity update of each particle, the PSO has memory so that the information of good solutions is retained by all individuals. Furthermore, the PSO has constructive cooperation between individuals so that individuals in the population share information between them. In the search of the PSO, the velocity update of each particle works for much of exploitation and less of exploration. Consequently, the diversity of the population decreases so that the search of the PSO results in the premature convergence. Much works have focused on the PSO to prevent the search process from the premature convergence [6–9]. In [9], the neighborhood of a particle is defined as the several closest individuals in every iteration so that a dynamic neighborhood is computationally intensive. In [8], a spatial extension is assigned to the particles and different collision strategies are applied to avoid crowding of the swarm. In [7], several neighborhood topologies have been examined for improving the PSO. In [6], a dynamically changing branching degree of the tree topology is introduced to improving the performance. In this paper, a hierarchical particle swarm optimization (HPSO) is proposed to increase the diversity of the population so that the premature convergence in the PSO approach is improved.

The rest of this paper is organized as follows. Section 2 describes the PSO in detail. Section 3 proposes the HPSO to improve the premature convergence in the PSO. In Section 4, the proposed HPSO is compared with some existing evolutionary algorithms using 10 benchmark functions with a low dimension and a high dimension. Finally, Section 5 draws conclusions about the proposed HPSO approach to solving optimization problems.

2. Particle Swarm Optimization

PSO is an evolutionary computation technique proposed by Kennedy and Eberhart. Its development was based on observations of the social behavior of animals such as bird flocking, fish schooling, and swarm theory. Like the GA, the PSO is initialized with a population of random solutions of the fitness function. The individuals $\underline{p}_k$, k = 1, 2, ..., m, in the population are called as particles. The PSO is an iterative method based on the search behavior of a swarm of m particles in a multidimensional search space. In each iteration, the velocity $\underline{v}_k$, $k \in \{1, 2, ..., m\}$ and the position $\underline{p}_k$, $k \in \{1, 2, ..., m\}$ of each particle are updated. According to the fitness values of the updated individuals, the personal best position $\underline{p}_k^{pbest}$, $k \in \{1, 2, ..., m\}$ of each particle and the global best position $\underline{p}^{gbest}$ among all the particles are updated. For the update of the velocities in the PSO, a particle $\underline{p}_k$ is influenced by its personal best position $\underline{p}_k^{pbest}$ and the global best position $\underline{p}^{gbest}$. Therefore, the PSO searches the global optimum solution by adjusting the trajectory of each particle toward its personal best position and the global best position. According to the above description about the PSO, the procedure of the PSO is described in the following:

Step 1. Initialize the PSO.

(a) Set the number of individuals (m), the number of iterations (G), and the constants for the PSO (c_1, c_2, w).

(b) Generate randomly initial individuals $\underline{p}_k$, k = 1, 2, ..., m, in the population.

(c) Generate randomly initial velocity vectors $\underline{v}_k$, k = 1, 2, ..., m.

Step 2. Calculate the fitness value of each individual and set initial $\underline{p}_k^{pbest}$, $\underline{f}_k^{pbest}$ and initial $\underline{p}^{gbest}$, $\underline{f}^{gbest}$ for the initial population.

(a) Set $f_k = fit(\underline{p}_k)$, k = 1, 2, ..., m, where $fit(\underline{p}_k)$ represents the fitness value of the individual $\underline{p}_k$ and then set $\underline{p}_k^{pbest} = \underline{p}_k$, $f_k^{pbest} = f_k$, k = 1, 2, ..., m.

(b) Find the index J of the individual with the best fitness value by $J = \arg \max_{k=1}^{m} f_k^{pbest}$ and then set $\underline{p}^{gbest} = \underline{p}_J^{pbest}$, $f^{gbest} = f_J^{pbest}$.

(c) Set iter = 1.

Step 3. Update $\underline{p}_k^{pbest}$, f_k^{pbest} and $\underline{p}^{gbest}$, f^{gbest}.

(a) Update $\underline{p}_k^{pbest}$, f_k^{pbest} in the following:

Calculate $f_k = fit(\underline{p}_k)$, k = 1, 2, ..., m, if $f_k > f_k^{pbest}$, then set $\underline{p}_k^{pbest} = \underline{p}_k$ and $f_k^{pbest} = f_k$.

(b) Update $\underline{p}^{gbest}$, f^{gbest} in the following:

If $f_k^{pbest} > f^{gbest}$, $k \in \{1, 2, ..., m\}$ then set $\underline{p}^{gbest} = \underline{p}_k^{pbest}$, $f^{gbest} = f_k^{pbest}$.

Step 4. Update the velocity vector $\underline{v}_k$ and the position vector $\underline{p}_k$ of each particle.

(a) Update the velocity vectors in the following:

$$\underline{v}_k = \underline{v}_k + c_1 \cdot rand() \cdot (\underline{p}_k^{pbest} - \underline{p}_k) + c_2 \cdot rand() \cdot (\underline{p}^{gbest} - \underline{p}_k),\ k = 1,2,\cdots, m, \quad (1)$$

where rand() is a uniformly distributed random number in [0,1].

(b) Update the position vectors in the following:

$$\underline{p}_k = \underline{p}_k + \underline{v}_k,\ k = 1,2,\cdots, m. \quad (2)$$

Step 5. Decrease the velocity vector by the constant $w \in [0,1]$.

$$\underline{v}_k = \underline{v}_k \cdot w,\ k = 1, 2, \cdots, m. \quad (3)$$

Step 6. iter = iter + 1, if iter > G then go to Step 7; otherwise go to Step 3.

Step 7. Based on the global best position $\underline{p}^{gbest}$ with the best fitness value f^{gbest}, the desired solution for the considered optimization problem can be determined.

Consequently, each individual $\underline{p}_k$ keeps track of its own best solution, which is associated with the best fitness value f_k^{pbest}, it has achieved so far in a vector $\underline{p}_k^{pbest}$. Furthermore, the best solution among all the individuals obtained so far in the population is kept track of as the vector $\underline{p}^{gbest}$ associated with the global best fitness value f^{gbest}. According to Step 4, each position vector $\underline{p}_k$ is assigned with a randomized velocity vector $\underline{v}_k$ according its own and its companions' flying experiences so that the position vector $\underline{p}_k$ searches around its personal best vector $\underline{p}_k^{pbest}$ and the global best vector $\underline{p}^{gbest}$. The adjustment toward $\underline{p}_k^{pbest}$ and $\underline{p}^{gbest}$ by Step 4 is conceptually similar to the crossover operator utilized by the GA. However, all the particles move toward the global best position $\underline{p}^{gbest}$ in Step 4 and the PSO has no the mutation operator. These probably lead to the premature convergence so that the obtained solution is trapped into the local maximum. In order to prevent the premature convergence, the HPSO is proposed in the next section.

3. Hierarchical Particle Swarm Optimization

In this section, the HPSO is proposed to improve the premature convergence in the PSO. In the HPSO, all particles are arranged in a hierarchy that defines a regular tree structure. The structure of the regular tree is determined by the height (h) and the branching degree (d) of the tree so that m nodes, s_k, k = 1, 2, …, m, are generated for constructing the regular tree, where $m = {d^h - 1}/{d - 1}$ is the number of nodes in the tree. In the initial population, m individuals, $\underline{p}_k$, k = 1, 2, ..., m, are generated initially and arranged orderly in the m nodes of the regular tree. That is, the individual $\underline{p}_k$, $k \in \{1, 2, ..., m\}$, is arranged in the node s_k, $k \in \{1, 2, \ldots, m\}$. For examples, a regular tree with h = 3 and d = 2 is shown in Figure 1. In Figure 1, seven nodes s_k, k = 1, 2, …, 7, are generated for constructing a tree and $\underline{p}_k$, k = 1, 2, ..., 7, are arranged in the nodes s_k, k = 1, 2, …, 7, respectively. In each iteration, the positions of two individuals in the tree are probably exchanged. For a individual $\underline{p}_k$ in the non-root node s_j, $j \in \{2, 3, ..., m\}$, the fitness value of the individual $\underline{p}_k$ is compared with that of the individual $\underline{p}_k^f$, where the individual $\underline{p}_k^f$ is in the parent node of s_j. If $fit(\underline{p}_k) > fit(\underline{p}_k^f)$, then $\underline{p}_k$ and $\underline{p}_k^f$ exchange their positions within the tree. Therefore, the individual $\underline{p}_k$ will move up one level in the tree. For examples, a regular tree with h = 3 and d = 2 is considered and the individuals in the population are arranged in the tree shown in Figure 2. Assume that the fitness values of the individuals $\underline{p}_i$, i = 1, 2, ..., 7, are 0.7, 0.1, 0.3, 0.2, 0.6, 0.5, 0.3, respec-

tively, and the individual $\underline{p}_k$ in the node s_3 is considered for changing its position in the tree. According to Figure 2, $\underline{p}_k$ and $\underline{p}_k^f$ are $\underline{p}_1$ and $\underline{p}_5$, respectively. Because fit($\underline{p}_1$) is larger than fit($\underline{p}_5$), $\underline{p}_1$ and $\underline{p}_5$ swap their positions within the tree so that the individuals are arranged in the tree shown in Figure 3. For the update of the velocities in HPSO, the velocity vectors of the particles $\underline{p}_k$, k = 1, 2, ..., m, are updated as follows:

$$\underline{v}_k = \underline{v}_k + c_1 \cdot \text{rand}() \cdot (\underline{p}_k^{\text{pbest}} - \underline{p}_k) + c_2 \cdot \text{rand}() \cdot (\underline{p}^f - \underline{p}_k) \quad (4)$$

where the parameter vector of $\underline{p}^f$ depends on the position of $\underline{p}_k$ in the tree. If $\underline{p}_k$ is in the root node, then $\underline{p}^f = \underline{p}^{\text{gbest}}$; otherwise $\underline{p}^f = \underline{p}_k^f$. For the update of the velocities in the HPSO, the particle $\underline{p}_k$ is influenced by $\underline{p}_k^{\text{pbest}}$ and $\underline{p}_k^f$ and the particle in the root node can indirectly influence all the other particles. Therefore, the HPSO searches the global optimum solution by adjusting the trajectory of each particle toward its personal best position and the position of the particle in its parent node. Consequently, the arrangement of the individuals leads to a different influence for the individuals at different positions. The changing arrangement of the individuals can help preserving diversity in the search. Furthermore, a mutation operator is added to keep the diversity of the population. According to the above description, the flowchart of the HPSO is shown in Figure 4 and the procedure of the HPSO for optimization problems is described in the following:

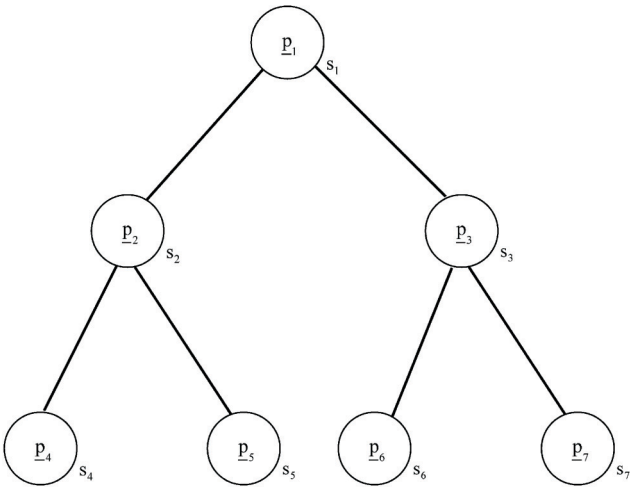

Figure 1. The individuals of the initial population arranged in a regular tree with h = 3 and d = 2.

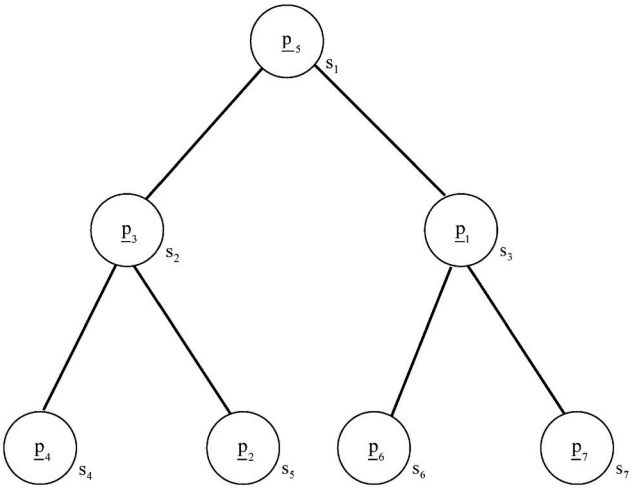

Figure 2. The individuals arranged in the tree for the example.

Step 1. Initialize the HPSO.

(a) Set the height of the tree (h), the branching degree of the tree (d), the maximum number of iterations (G), the constants for the HPSO (c_1, c_2, w, r) and the mutation rate (P_m).

(b) Generate randomly the initial population P in the nodes of the tree.

Each individual of the population is expressed as follows:

$$\underline{p}_k = [x_1^k \;\; x_2^k \;\; \cdots \;\; x_D^k], \;\; k \in \{1, 2, \cdots, m\} \quad (5)$$

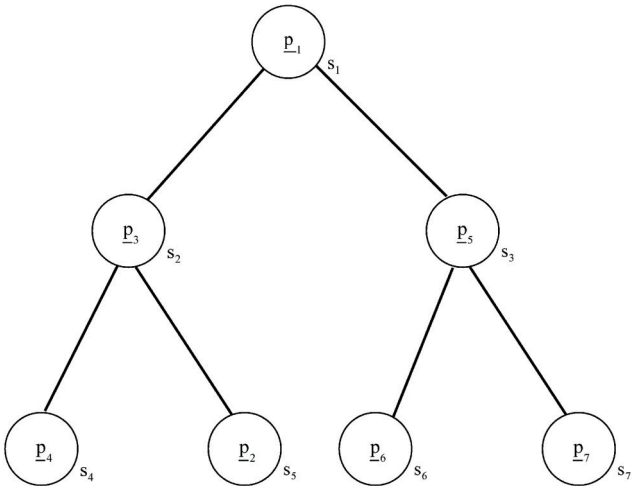

Figure 3. After the position exchange, the individuals arranged in the tree for the example.

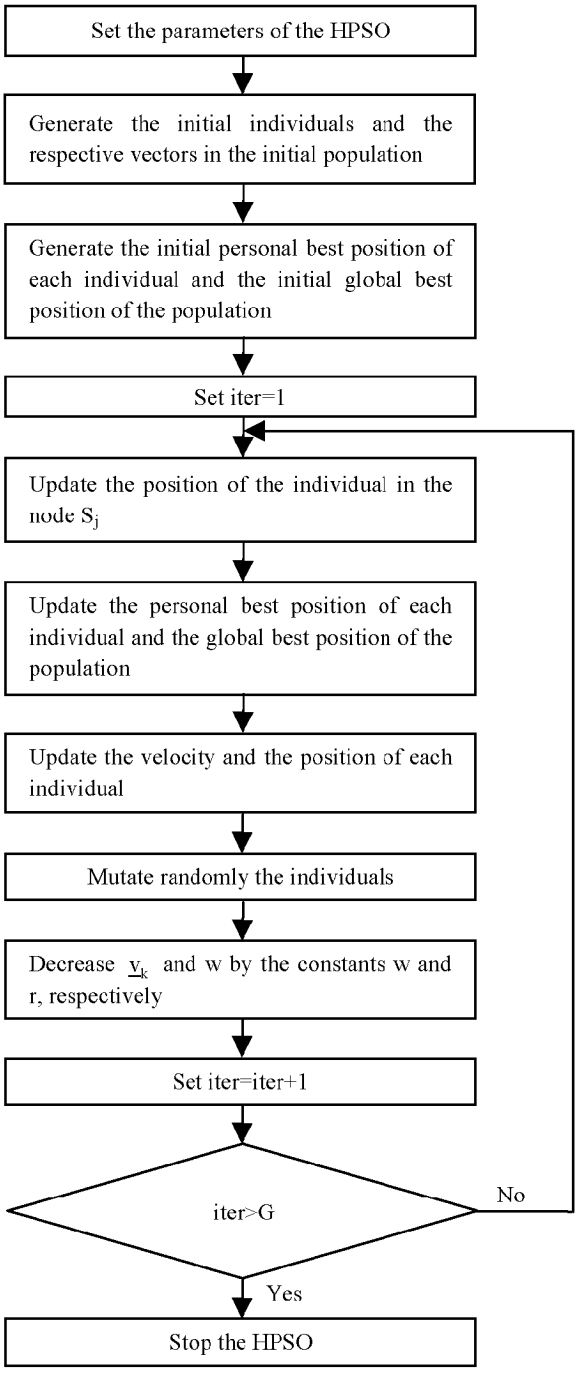

Figure 4. The flowchart of the HPSO.

where x_i^k, $i \in \{1, 2, ..., D\}$, is randomly generated as follows:

$$x_i^k = x_i^{min} + (x_i^{max} - x_i^{min}) \cdot rand() \tag{6}$$

where the range of the parameter x_i^k is defined as $[x_i^{min}, x_i^{max}]$.

(c) Generate randomly initial velocity vectors $\underline{v}_k$, k = 1, 2, ..., m.

Each velocity vector is expressed as follows:

$$\underline{v}_k = [v_1^k \ v_2^k \ \cdots \ v_D^k], k \in \{1, 2, \cdots, m\} \tag{7}$$

where v_i^k, $i \in \{1, 2, ..., D\}$, is randomly generated as follows:

$$v_i^k = \frac{x_i^{max} - x_i^{min}}{20} \cdot rand() \tag{8}$$

Step 2. Calculate the fitness value of each individual and set initial $\underline{p}_k^{pbest}$, f_k^{pbest} for each individual and initial $\underline{p}^{gbest}$, f^{gbest} for the population.

(a) Set $f_k = fit(\underline{p}_k)$, k = 1, 2, ..., m, and then set $f_k^{pbest} = f_k$, $\underline{p}_k^{pbest} = \underline{p}_k$, k = 1, 2, ..., m.

(b) Find the index J of the individual with the best fitness by $J = \arg \max_{k=1}^{m} f_k^{pbest}$. Set $f^{gbest} = f_J^{pbest}$, $\underline{p}^{gbest} = \underline{p}_J^{pbest}$

(c) Set iter = 1.

Step 3. Update the position of the individual in the node s_j, where $j = ((iter - 1) \bmod (m - 1)) + 2 \in \{2, 3, ..., m\}$. Here, (iter – 1) mod (m – 1) is the modulus after (iter – 1) is divided by (m – 1). Assume the individual $\underline{p}_k$ is in the node s_j and the individual $\underline{p}_k^f$ is in the parent node of s_j. If $fit(\underline{p}_k) > fit(\underline{p}_k^f)$, then $\underline{p}_k$ and $\underline{p}_k^f$ swap their positions within the tree.

Step 4. Update $\underline{p}_k^{pbest}$, f_k^{pbest} and $\underline{p}^{gbest}$, f^{gbest}.

(a) $f_k = fit(\underline{p}_k)$, k = 1, 2, ..., m. If $f_k > f_k^{pbest}$, then set $\underline{p}_k^{pbest} = \underline{p}_k$ and $f_k^{pbest} = f_k$.

(b) If $f_k^{pbest} > f^{gbest}$, $k \in \{1, 2, ..., m\}$, then set $\underline{p}^{gbest} = \underline{p}_k^{pbest}$, $f^{gbest} = f_k^{pbest}$.

Step 5. Update the velocity vectors $\underline{v}_k$, k = 1, 2, ..., m, and the parameter vectors $\underline{p}_k$, k = 1, 2, ..., m.

$$\begin{aligned} (a)\ \underline{v}_k &= \underline{v}_k + c_1 \cdot rand() \cdot (\underline{p}_k^{pbest} - \underline{p}_k) \\ &\quad + c_2 \cdot rand() \cdot (\underline{p}^f - \underline{p}_k), \ k = 1, 2, \cdots, m \end{aligned} \tag{9}$$

where $\underline{p}^f$ depends on the position of the individual

$\underline{p}_k$ in the tree. If $\underline{p}_k$ is in the root node of the tree then $\underline{p}^f = \underline{p}^{gbest}$; otherwise $\underline{p}^f = \underline{p}_k^f$.

(b) $\underline{p}_k = \underline{p}_k + \underline{v}_k$, $k = 1, 2, \cdots, m$ (10)

Step 6. Mutate randomly the individual $\underline{p}_k = [x_1^k \; x_2^k \cdots x_D^k]$, $k \in \{1, 2, ..., m\}$.
If $P_m \geq$ rand(), then $x_{i^*}^k = x_{i^*}^{max} - (x_{i^*}^k - x_{i^*}^{min})$, where $i^* = \text{round}(D \cdot \text{rand}() + 0.5)$. round($D \cdot$ rand() + 0.5) rounds($D \cdot$ rand() + 0.5) to the nearest integer.

Step 7. Decrease $\underline{v}_k$ and w by the constants $w \in [0,1]$ and $r \in [0,1]$, respectively.

(a) $\underline{v}_k = \underline{v}_k \cdot w$, $k = 1, 2, \cdots, m$ (11)

(b) If $w \geq 0.1$ then $w = w \cdot r$; otherwise $w = 0.1$.

Step 8. iter = iter + 1, if iter > G then go to Step 9; otherwise go to Step 3.

Step 9. Based on $\underline{p}^{gbest}$ with the best fitness f^{gbest}, the best solution for the considered optimization problem can be determined.

4. Simulations

In this section, 10 benchmark functions including unimodal and multimodal functions [10] are employed to examine the efficiency of the proposed HPSO for optimization problems. The test function, parameter domain, and global optimum for each benchmark function are listed in Table 1. In order to examine the proposed HPSO approach to optimization problems with a low dimension and a high dimension, the dimensions of the 10 benchmark functions are set to D = 10 and D = 100 in the experiments, respectively. The initial conditions for the

Table 1. The test function, parameter domain, and global optimum for each benchmark function

Test function	x_i domain	Optimum
$f_1 = -\sum_{i=1}^{D}\left[\sin(x_i) + \sin(\frac{2x_i}{3})\right]$	[3,13]	1.21598D (max)
$f_2 = -\sum_{i=1}^{D-1}\left[\sin(x_i + x_{i+1}) + \sin(\frac{2x_i x_{i+1}}{3})\right]$	[3,13]	≈ 2D (max)
$f_3 = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12,5.12]	0 (min)
$f_4 = \sum_{i=1}^{D} x_i^2$	[-5.12,5.12]	0 (min)
$f_5 = \sum_{i=1}^{D}\left(x_i \sin(10\pi x_i)\right)$	[-1,2]	1.85D (max)
$f_6 = \sum_{i=1}^{D}\left\|\frac{\sin(10x_i\pi)}{10x_i\pi}\right\|$	[-0.5,0.5]	0 (min)
$f_7 = 20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}} - e^{\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}}$	[-30,30]	0 (min)
$f_8 = 418.9828D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	0 (min)
$f_9 = \sum_{i=1}^{D-1}\left[\, 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\,\right]$	[-5.12,5.12]	0 (min)
$f_{10} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0 (min)

proposed HPSO method to each benchmark function are given in the following: the height of the tree: h = 3, the branching degree of the tree: d = 4, the mutation rate: P_m = 0.1, the constants for the HPSO: (c_1, c_2, w, r) = (2, 2, 0.9, 0.95) and the maximum number of iterations for each benchmark function is listed in Table 2. The results of the proposed HPSO approach to each benchmark function using 30 independent runs for D = 10 and D = 100 are shown in Tables 3 and 4, respectively. Furthermore, Tables 5 and 6 compare our results of the proposed HPSO approach with the results obtained by the PSO approach [11] for each benchmark function with D = 10 and D = 100, respectively. According to Tables 5 and 6, it reveals that the PSO approach method might lead to the earlier convergence so that the result of the PSO approach is trapped into the local optimum solution owing

Table 2. Number of the iterations for each benchmark function

D = 10		D = 100	
Test function	Number of the iterations	Test function	Number of the iterations
f_1	100	f_1	1000
f_2	500	f_2	5000
f_3	100	f_3	1000
f_4	100	f_4	1000
f_5	500	f_5	5000
f_6	100	f_6	1000
f_7	100	f_7	1000
f_8	500	f_8	5000
f_9	100	f_9	1000
f_{10}	100	f_{10}	1000

Table 3. Results of the proposed HPSO approach to each benchmark function using 30 independent runs for D = 10

Test function	Best function value	Worst function value	Mean function value	Optimal function value
f_1	12.1598	12.1591	12.1597	12.1598
f_2	17.9436	15.5322	16.8107	≈ 20
f_3	0	0.002	9.524×10^{-5}	0
f_4	8.785×10^{-29}	1.8×10^{-19}	1.2×10^{-20}	0
f_5	18.5027	18.5003	18.5025	18.5
f_6	3.8982×10^{-16}	3.8982×10^{-16}	3.8982×10^{-16}	0
f_7	2.3404×10^{-13}	1.0019×10^{-8}	1.3208×10^{-9}	0
f_8	0.0715	247.032	74.0719	0
f_9	0.3932	5.5504	3.9732	0
f_{10}	0	0	0	0

Table 4. Results of the proposed HPSO approach to each benchmark function using 30 independent runs for D = 100

Test function	Best function value	Worst function value	Mean function value	Optimal function value
f_1	121.5963	121.5917	121.5949	121.598
f_2	181.8365	160.8262	171.8359	≈ 200
f_3	0	3.1631×10^{-5}	1.0929×10^{-6}	0
f_4	3.8393×10^{-25}	2.6131×10^{-21}	2.8784×10^{-22}	0
f_5	185.0257	183.2530	184.7754	185
f_6	3.8982×10^{-15}	3.8982×10^{-15}	3.8982×10^{-15}	0
f_7	1.2027×10^{-11}	6.1932×10^{-11}	3.2233×10^{-11}	0
f_8	5506.1	9491.4	7875.4	0
f_9	93.2766	95.7135	95.2838	0
f_{10}	0	1.6331×10^{-13}	7.2127×10^{-15}	0

Table 5. Comparison between our results of the proposed HPSO and the results obtained by the PSO method for each benchmark function with D = 10

Test function	HPSO			PSO		
	Best function value	Worst function value	Mean function value	Best function value	Worst function value	Mean function value
f_1	12.1598	12.1591	12.1597	11.6227	7.3027	10.2999
f_2	17.9436	15.5322	16.8107	11.7136	7.1255	9.5543
f_3	0	0.002	9.524×10^{-5}	43.7737	93.1691	68.1531
f_4	8.785×10^{-29}	1.8×10^{-19}	1.2×10^{-20}	1.3020	8.2790	3.4799
f_5	18.5027	18.5003	18.5025	7.5354	3.2205	5.5299
f_6	3.8982×10^{-16}	3.8982×10^{-16}	3.8982×10^{-16}	0.0113	0.0125	0.0118
f_7	2.3404×10^{-13}	1.0019×10^{-8}	1.3208×10^{-9}	2.2658	5.0638	3.5164
f_8	0.0715	247.032	74.0719	3535	3872	3741
f_9	0.3932	5.5504	3.9732	59.1235	1182.124	443.85
f_{10}	0	0	0	8.8	29.28	16.992

Table 6. Comparison between our results of the proposed HPSO and the results obtained by the PSO method for each benchmark function with D = 100

Test function	HPSO			PSO		
	Best function value	Worst function value	Mean function value	Best function value	Worst function value	Mean function value
f_1	121.5963	121.5917	121.5949	79.8338	57.0058	67.6352
f_2	181.8365	160.8262	171.8359	53.7467	27.2423	35.6987
f_3	0	3.1631×10^{-5}	1.0929×10^{-6}	990.248	1188.882	1091.34
f_4	3.8393×10^{-25}	2.6131×10^{-21}	2.8784×10^{-22}	173.534	318.2602	247.663
f_5	185.0257	183.2530	184.7754	25.2286	10.0302	16.5484
f_6	3.8982×10^{-15}	3.8982×10^{-15}	3.8982×10^{-15}	0.1238	0.125	0.1247
f_7	1.2027×10^{-11}	6.1932×10^{-11}	3.2233×10^{-11}	7.9584	9.6176	8.7934
f_8	5506.1	9491.4	7875.4	37040	40737	38509
f_9	93.2766	95.7135	95.2838	70562	215022	129307
f_{10}	0	1.6331×10^{-13}	7.2127×10^{-15}	292.8	497.6	395.178

to the lack of swarm's diversity. However, the proposed HPSO has more diversity such that it is hard to be trapped into the local optimum owing to having more searching choices for the particle swarm. Tables 7 and 8 compare our results of the proposed HPSO approach with the results obtained by several evolutionary algorithms [4] for each benchmark function with D = 10 and D = 100, respecitvely. Tables 7 and 8 show that the proposed HPSO outperforms these evolutionary algorithms. Consequently, the proposed HPSO is an effective tool for solving optimization problems.

5. Conclusion

In this paper, a hierarchical particle swarm is proposed to overcome the premature convergence in the PSO approach to optimization problems. In order to improving the premature convergence, the particles are arranged in a regular tree and move up or down in the tree according to their fitness values. The velocity update of

Table 7. Mean function values obtained by the proposed HPSO method and other evolutionary algorithms to each benchmark function using 30 independent runs for D = 10

Test function	HPSO	IEA	OEGA	UEGA	TEGA	BLXGA	OGA
f_1	12.1597	12.116	12.119	12.110	12.113	12.150	12.109
f_2	16.8107	15.32	16.19	16.39	15.89	16.52	15.24
f_3	9.524×10^{-5}	15.42	8.94	9.57	11.63	6.48	17.62
f_4	1.2×10^{-20}	0.0003	0.0004	0.0003	0.0006	0.0092	0.0003
f_5	18.5025	14.60	15.82	15.54	15.24	15.85	14.01
f_6	3.8982×10^{-16}	0.054	0.035	0.040	0.039	0.017	0.072
f_7	1.3208×10^{-9}	1.00	0.23	0.36	0.75	1.93	1.70
f_8	74.0719	667.4	848.1	1859.5	1132.0	714.7	584.2
f_9	3.9732	116.44	41.39	39.56	23.00	22.63	94.57
f_{10}	0	0.999	1.001	1.030	1.002	1.030	1.002

Table 8. Mean function values obtained by the proposed HPSO method and other evolutionary algorithms to each benchmark function using 30 independent runs for D = 100

Test function	HPSO	IEA	OEGA	UEGA	TEGA	BLXGA	OGA
f_1	121.5949	120.44	90.52	89.36	87.63	88.9	116.71
f_2	171.8359	153.15	94.59	89.49	94.27	74.47	139.71
f_3	1.0929×10^{-6}	213.46	795.75	819.65	791.65	763.42	367.51
f_4	2.8784×10^{-22}	1.60	145.52	151.43	154.97	67.89	14.96
f_5	184.7754	131.31	76.72	74.50	76.06	51.53	115.48
f_6	3.8982×10^{-15}	0.65	3.86	3.94	3.82	5.45	1.63
f_7	3.2233×10^{-11}	3.69	16.93	16.83	16.97	14.35	9.47
f_8	7875.4	8011	24645	24160	24723	24794	12294
f_9	95.2838	2081	96556	89514	97990	16086	5282
f_{10}	7.2127×10^{-15}	32.86	511.93	1002	537.09	237.84	48.25

each particle depends on the position of the particle in the tree. Furthermore, a mutation operator is added in the HPSO approach. Consequently, the diversity of the population in the proposed HPSO will increase so that the HPSO approach has a much opportunity of finding the global optimum solution. According to the simulation results for 10 benchmark functions, it is clear that the proposed HPSO approach is superior to the other evolutionary algorithms in the ability to finding the global optimum solution.

Acknowledgement

This research was supported in part by the National Science Council of the Republic of China under contract NSC 98-2221-E-274-009.

References

[1] Davis, L., *Handbook of genetic algorithms,* New York: Van Nostrand Reinhold (1991).

[2] Eberhart, R. and Kennedy, J., "A New Optimizer Using Particle Swarm Theory," *Proc. Int. Sym. Micro Machine and Human Science,* Nagoya Japan, pp. 39–43 (1995).

[3] Kennedy, J. and Eberhart, R., "Particle Swarm Optimization," *Proc. IEEE Int. Conf. Neural Networks,* Perth, Australia, pp. 1942–1948 (1995).

[4] Eshelman, L. J. and Schaffer, J. D., *Real-coded genetic algorithms and interval schemata,* Foundation of Ge-

netic Algorithm-2. L. D. Whitley, ED. San Mateo, CA: Morgan Kaufmann (1993).

[5] Zhang, Q. and Leung, Y. W., "An Orthogonal Genetic Algorithm for Multimedia Multicast Routine," *IEEE Transactions Evolutionary Computation,* Vol. 3, pp. 53–62 (1999).

[6] Janson, S. and Middendorf, M., "A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics,* Vol. 35, pp. 1272–1282 (2005).

[7] Jian, W., Xue, Y. C. and Qian, J. X., "Improved Particles Swarm Optimization Algorithms Study Based on the Neighborhoods Topologies," *The 30th Annual Conference of the IEEE Industrial Electronics Society,* Busan Korea, pp. 2192–2196 (2004).

[8] Krink, T., Vesterstrom, J. S. and Riget, J., "Particle Swarm Optimization with Spatial Particle Extension," *Proc. IEEE Congr. Evolutionary Computation* (CEC 2002), pp. 1474–1479 (2002).

[9] Suganthan, P. N., "Particle Swarm Optimizer with Neighborhood Operator," *in Proc. Congr. Evolutionary Computation* (CEC 1999), pp. 1958–1962 (1999).

[10] Ho, S. Y., Shu, L. S. and Chen, J. H., "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," *IEEE Transactions on Evolutionary Computation,* Vol. 8, pp. 522–541 (2004).

[11] Lin, H. S., *Design of a novel orthogonal particle swarm optimization*. Master thesis, Feng Chia University, Taiwan, Republic of China (2004).

Manuscript Received: Jan. 22, 2008
Accepted: Sep. 22, 2008