Hierarchical Particle Swarm Optimization Algorithm for Multimodal Function Optimization

Qin Gao[1], Yi Zhong[1,a], Xinjuan Zheng[1]

1:School of Information Engineering,Wuhan University of Technology, Wuhan,Hubei,430070,China
E-mails: zhongyi@whut.edu.cn; Tel: +86 130-3513-0763

Corresponding author is Yi Zhong

Abstract
In this paper, we propose a Hierarchical Particle Swarm Optimization (HPSO) algorithm model for multi- modal function optimization. All particles will be classified into several groups, these groups operated at two levels: one level is to find location of global optima clusters with its particles; the other is to exactly distinguish multiple global optima in clusters. By this mechanism, algorithm can explore the search space at different level of scales simultaneously; all the global optima can be detected effectively. In addition, the algorithm is easy to be controlled; only two parameters are needed to be predefined before the running of algorithm and algorithm's performance is not sensitive to the influence caused by these two parameters. We tested our algorithm on several benchmark functions and compared it with some well-known methods for these functions. Experimental results showed that our algorithm could achieve an overall good performance. We also analyzed the effect of the parameters for our algorithm performance at the end of our paper.
Keywords: HIERARCHICAL PARTICLE SWARM OPTIMIZATION (HPSO), MULTIMODAL FUNCTION OPTIMIZATION, GA; PSO

1. Introduction

The research of Multimodal Optimization Algorithm has been developed widely for many years, especially in the field of Engineering Design. It is different from other traditional optimization algorithms, which are designed to find single global optima; multimodal optimization algorithm is designed with the function of finding the global optima as many as possible in searching space. Multimodal optimization algorithm has the practical applications in many areas such as Economics [1], and Path Plan [2]. For example, when we are making a plan, we usually want to find out all the solutions to satisfy the requirements first and then select the best one among them manually according to other criterions. The first step of this solution can be converted to a typical multimodal optimization problem; however, such problems are always demonstrated to be very complex and can hardly be solved via the traditional method.

Evolutionary Algorithms (EA) such as Genetic Algorithm(GA) and Particle Swarm Optimization(PSO) algorithm has gained popularity increasingly and has been successfully applied to all kinds of optimization problems in the past decades years [3,4,5]. However, Pena et.al [6] pointed out that classical EAs are ineffective for solving multimodal problems since they are converge to a single global solution, when there are several global solutions aroused from a same population, no more significant selective preference could be distinguished between

them, and thus, only one global solution can be found. In order to make up the defect of classical evolution algorithm, researchers proposed many methods: in [3] and [7] authors modified the selection procedure of classical GA; in [8], the author used multiple populations; Li applied species idea to the PSO algorithm [5, 9] and Swarm Intelligence [10].

While these modified EAs achieved great success for solving multimodal function optimization problem, there are still many technical challenges. One of which researchers often encountered is that how we can find all the global optima of multimodal functions effectively and efficiently. Their global optima are usually unevenly distributed: some optima close to each other in a small region, while others depart from each other far away. Among evolutionary algorithms mentioned above, some previous researches indicated that PSO algorithm can locate the region of optimum faster than GA, but once in this region PSO algorithm progressed slowly [1]. This is to say, PSO algorithm is more efficient than GA in the early stage of exploring, however, it is not good at searching at a finer grain. If there are two global optima in a small region, GA should be more effective to distinguish them than PSO algorithm should do. In order to enhance the ability of solving multimodal function optimization problems of PSO algorithm, researchers proposed some new methods. It showed us [5] the evidence that Species Particle Swarm Optimization (SPSO) algorithm can find all the global optima with less function evaluation numbers than GA can do for some functions. But he only tested SPSO algorithm on the functions which global optima are relatively evenly distributed, for some extremely conditions, such as Shubert function; SPSO algorithm cannot effectively find all global optima. Petropoulos's hybrid algorithm [1] is effective in many cases, but it seems too complicated.

In this paper, we proposed a hierarchy PSO (HPSO) algorithm; this algorithm can explore the search space in different level of scales simultaneously. All particles will be divided into several groups dynamically; groups can be classified into two levels: high level and low level. Particles which belong to high level groups search the whole space. In high level groups, some particles can select the particles around them to make up a low level group. By this method, some particles search at coarse grain while others search at fine grain. Basic particle swarm optimization algorithm is applied to these two levels. In fact, such hierarchical structure is very popular in nature. For example: tread milling is the unified principle applied to all levels of motility within the cell, from the action of a single action filament to the movement of a large motile cell coordinating all of its components to protrude in the front and retract in the rear [11]. By this mechanism, cell can move to proper position of our brain and life can be continued. Our experimental results also proved that this idea can enable PSO algorithm to solve multimodal optimization problem.

In section 2, we give an overview of GA and PSO algorithm for multimodal optimization. Then in section 3, we describe the implementation of HPSO. In section 4, we evaluate HPSO's performance on several benchmarks problems. In section 5, we give a discussion on the parameters of HPSO. Conclusion will be given in section 6.

2. Materials and Methods

The evolutionary computation community has shown a significant interest in multimodal function optimization for many years and most of the researches are based on genetic algorithms.

Ando et.al [8] proposed a GA based model called Adaptive Isolation Model (AIM) for multimodal optimization, they used a data clustering algorithm to detect clusters in GA population, which identifies the attractors in the fitness landscape and then clusters will optimized independently. This multi-population idea is also used by other algorithms like ANS [12]. Li et.al [3] proposed an algorithm called Species Conserving Genetic Algorithm (SCGA) which modified the GA selection procedure, in SCGA algorithm population is divided into several species according to their similarity in term of Euclidean distance, each of these species is built around a dominating individual called the species seed. Species will be in each selection iteration reconstruction.

Li [5] applied specie idea to particle swarm optimization algorithm and invented Species Particle Swarm Optimization (SPSO) algorithm. In this method, some better solution is selected as seed in each iteration; the particles around the seed are chosen to make up the population of the seed. This algorithm is proved to converge faster than SCGA. However, Li only tested his algorithm on a small test suite. In fact, since the parameter radius is fixed during the evolution, it would be hard to distinguish two global optima if they are similar to each other.

Petropoulos et.al[1] presented a hybrid PSO algorithm to find all global minimizers of a multimodal function, this approach transforms objective function by deflection, stretching and repulsion techniques at each detected minimizer, thus, the detected minimizers can be avoided to be detected again, all the global optima can be found.

Our algorithm is somewhat liking the SPSO algorithm, it is also based on particle swarm optimization algorithm and also uses multi-population idea. The big difference is that we use two level scales and dynamical parameters, ensure that the algorithm can effectively find all global optima if its efficiency is equal to the basic PSO in unmoral conditions.

3. Hierarchy Particle Swarm Optimization

A. Basic PSO

The particle swarm optimization (PSO) algorithm was firstly proposed by Eberhart and Kennedy [13]. In the past several years, PSO has grown fast and it has been investigated from various perspectives [14]. Like other stochastic optimization algorithms. PSO begins with a population of randomly distributed particles, and then in the following iterations, particles update their positions based on predefined rules. To solve a d-dimensional multimodal function optimization problem, we can use N particles $(X_1, X_2, ..., X_N)$, each particle is a d-dimensional vector and can be represented as $X_i = (x_i^1, x_i^2, ..., x_i^d)$, and the velocity of this particle is denoted as $V_i = (v_i^1, v_i^2, ..., v_i^d)$. The best solution this particle found so far is denoted as $P_i = (p_i^1, p_i^2, ..., p_i^d)$. We use $P_g = (p_g^1, p_g^2, ..., p_g^d)$ to represent the best solution of all N particles. At $(K+1)th$ iteration, each particle updates its position according to following equations:

$$V_i = (K+1) = \omega V_i(K) + c_1 r_1 (P_i(K) - X_i(K)) + c_2 r_2 (P_g(K) - X_i(K))$$
$$X_i = (K+1) = X_i(K) + V_i(K+1) \quad (1)$$

Where ω is called inertia weight, c_1 and c_2 are learning factors, r_1 and r_2 are random numbers uniformly distributed in [0, 1].

B. Hierarchy PSO

Hierarchy particle swarm optimization is based on the basic PSO. As described in Figure. (1), the procedure of our algorithm can be divided into four steps: initialization, creating high level groups, creating low level groups and updating particles' positions. At the first step, we initialize the particles' positions randomly and assign initial values to parameters. At the second step, some particles are selected as seeds; a high level group will be created around each seed. In the third step, some smaller groups which called low level groups are created inside the high level groups. At the last step, all particles update their positions according to Equation (1). The last three steps will be done iteratively until stop criteria is satisfied. Details of each step will be described in following sections.

We list some important parameters in Table 1. Among them, only two parameters SpecMin and SpecMax are need to be predefined by user, others are adaptively tuned while algorithm running. Effect of the two parameters for the algorithm's performance will be discussed in Section 5.

Initialization. Like other evolutionary algorithms, HSPO has to initialize its population before running the algorithm. For some evolutionary algorithms such as GA, initializing population according to the properties of objective function can greatly improve algorithm's performance, however, such initialization algorithm itself is not easy and it is not suitable for general algorithm. In this paper, we use the simplest method: putting particles in search space randomly.

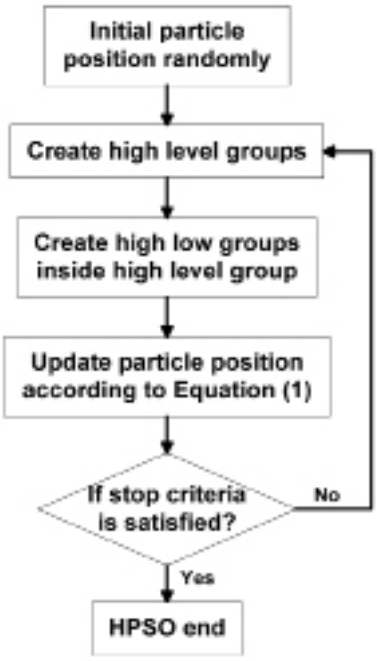

Figure 1. Flowchart of HPSO

Table 1. Parameter for HPSO Algorithm

Parameter	Comments
PopSize	particle number in population
SpecMin	Minimum number of high level groups
SpecMax	Maximum number of high level groups
r_l	radius of high level groups
dev_i	standard deviation of high level group *i*
δ_{fi}	maximum fitness difference between seed of high level group *i* and its low level group seed
δ_{di}	minimum distance difference between seed of high level group *i* and its low level group seed
Vel_{max}	Maximum velocity for particle

Before randomly putting all particles in search space, we need to know how many particles (PopSize) are used to search the space. For most evolution algorithms, the number of particles is predefined by user before running the algorithm. Usually, for basic PSO algorithm, this value is set from 20 to 50. In our algorithm, we divide all particles into several groups, if there are many groups but particles are insufficient, some groups will have no particles and algorithm will fail.

From our experience, it is properly to ensure each group at least has 10 particles, the number of groups decided by the property of objective function, if the objective function has many local optima, it is better use a large population size. Obviously, large population size will make our algorithm running slowly, however it also make algorithm more effective. Actually, as will show in Section 6, converge speed will not dramatically change with population size increasing.

SpecMin and SpecMax are the other two parameters which needed to be predefined. Most of researchers who apply species idea to evolution algorithm muse species distance r_a to control their algorithms. This method has one disadvantage: the minimum distance of two global optima is not easy to know, so it maybe needs to try many times to get a proper value for each objective function. If r_a is too large, many optima will locate into same group, if r_a is too small, some groups will have few particles.

Using SpecMin and SpecMax to dynamically control parameter r_a, algorithm will become more robust. SpecMin and SpecMax represent how many high level groups, in other words, how many global optima clusters in search space. One optima cluster may have one or several global optima.

For example, two dimensional Shubert function have 18 global optima in 9 clusters, each cluster have two optima close to each other. In our algorithm, particles in low level groups will distinguish the two optima in one cluster. For Shubert function, we can set $SpecMin = 10$ and $SpecMax = 25$ or $SpecMin = 8$ and $SpecMax$ 15 or some other values. As we can see in Section 6, algorithm is not so sensitivity to the parameter SpecMin and SpecMax, since there are two level groups in population, while high level groups is relatively stable during searching, low level groups changed dynamically.

As the last step of initialization procedure, we need to set an initial value to r_a for creating high level groups. We use the following equation:

$$r_a = \frac{\frac{1}{2}\sqrt{\sum_{k=1}^{d}(x_k^u - x_k^l)^2}}{\sqrt[d]{SpecMax}} \quad (2)$$

Where x_k^u and x_k^l are respectively the upper and lower bounds of the d dimensional search space. This approach is suggested in [3], Li also suggested that this approach can only be used in cases where global optima are evenly distributed over feasible region. In our algorithm, r_a is changed adaptively in each iteration, so when we test the algorithm on some functions which global optima are unevenly distributed, the initial value also works well.

Create High Level Groups. The duty of particles in high level groups is to locating the global optima clusters. Each high level group has a seed; the seed is the best solution in this group. Each seed can choose its members based on their similarity to create a high level group. Euclidean distance r between two particles $X_i = (x_i^1, x_i^2, ..., x_i^d)$ and $X_j = (x_j^1, x_j^2, ..., x_j^d)$ is used to define the similarity, smaller distance means more similarity. Distance r is defined by:

$$r = d(X_I, X_j) = \sqrt{\sum_{k=1}^{d}(x_i^k - x_j^k)^2} \quad (3)$$

The procedure for creating a high level group is described in Figure. (2). To some extent, one high level group indicates that there are one or several global optima around the seed. Ideally, if all the global optima become seeds, all global optima would be found. However, it is not easy to do that, since all the high level groups share the same radius threshold value ra, but optima are not evenly distributed, inevitably, some high level groups will have several optima. In our algorithm, we do not hope high level groups found all global optima, their duty is to finding an area where may have one or several optima, then low level groups will detect all the global optima one by one.

Create Low Level Groups. It is natural to think that dividing the group into more smaller one can increase the possibility to distinguish two global optima in a small region, however, there are at least two difficulties: one is that we do not know how much the two close each other, so we do not know how small our low level groups should be; the other one is where to locate the low level groups. Of course we can do the same thing as we did to create high level groups, select a particle in high level group as seed, and then choose the particles around this seed to create a low level group. However, if do like that, all the particles will belong to low level groups and algorithm will converge to local optima immediately. We hope some

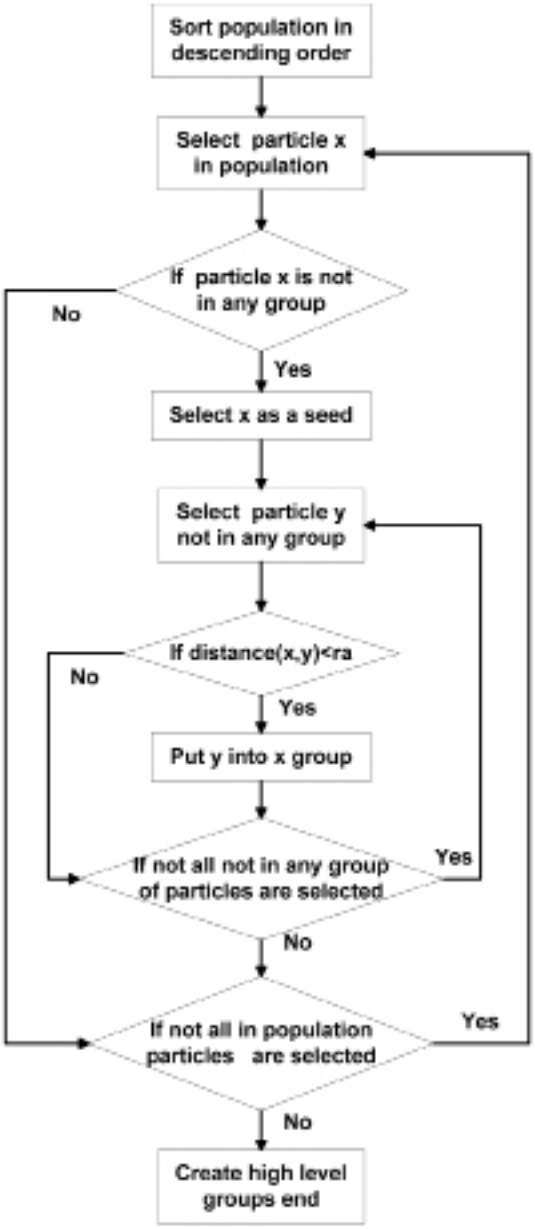

Figure 2. Create high level groups

particles belong to high level groups; others belong to low level groups, so algorithm can explore the search space at different level of scales.

For creating low level groups, we introduce two parameters δ_{fi} and δ_{di}. δ_{fi} represent the maximum fitness difference between high level group seed and low level group seed; δ_{di} represent the minimum distance difference between high level group seed and low level group seed. If a particle's fitness is close to the seed of high level group (the fitness difference not larger than δ_{fi}) and its position is not close to the seed of high level group (the distance difference not less than δ_{di}), we choose it as a low level group seed, it means that the particle has the potential to be global optima. The procedure for creating low level groups is described as follow:

if ((seed fitness - particle i fitness) $< \delta_{fi}$)&&
distance (seed, particle i) $> \delta_{di}$)
{
select particle i as low level group seed;
for particles k not in any group
{
if (distance(i, k)) < (distance(seed, particle i)/2)
put particle k into low level group;
}
}

Here $\delta_{di} = r_a / (10 + \sqrt{\text{int}eration})$, $\delta_{fi} = 2 * dev\sqrt{2}$. dev represents the standard deviation of high level group. δ_{di} cannot be too small, otherwise it will make too many low level groups and algorithm will converge very slowly. In our algorithm, the value of δ_{di} is decreasing with iteration, by this mechanism, even though we did not set properly r_a value at beginning, δ_{di} will approach to the minimum distance of global optima gradually.

Update Particle Position. The last procedure is updating particle's positions. Here we apply basic particle optimization algorithm to both high level and low level groups. The parameters are set as following:

$$V_i(K+1) = 0.729844 * V_i(K) + 2.05 * (P_i(K) - X_i(K)) + 2.05 * (P_g(K) - X_i(K)) \quad (4)$$
$$X_i(K+1) = X_i(K) + V_i(K+1)$$

These parameters are suggested by [15]. P_g is to the seed of groups which particle is belong to. As explained in above section, r_a initial value is only suitable for specific conditions, so we have to adjust it each iteration. In this paper, we use SpecMin and SpecMax to control the algorithm, the value of r_a is also changed according to these two parameters:

if(high level group number < SpecMin)
$r_a = r_a * 1.1$
if(high level group number > SpecMax)
$r_a = r_a / 1.1$

4. Experiments

In this section, the performance of proposed method is tested on unimodal and multimodal benchmarks. In our experiments, each test will run 50 times, mean value and standard deviation of function evaluations is recorded.

A. Low Dimensional Multimodal Functions.

We test our algorithm on 10 benchmark problem, experimental results are listed in Table 2, Table 3 and Table 4 We use ω_i to represent the fitness of global optima we have found, use gi to represent the fitness of known global optima, if ω_i satisfy following con-

ditions, we said algorithm have found all global optima successfully.

for all known global optima i

$$(abs(\omega_i - g_i) < \varepsilon) == TRUE$$

On each objective function, we got a 100% success rate to find all global optima with accuracy $\varepsilon = 0.00001$. In all experiments, we set $SpecMin = 5$ and $SpecMax = 10$.

$$F_1(x) = \sin^6(5\pi x)$$

$$F_2(x) = \exp(-2\log(2)\cdot(\frac{x-0.1}{0.8})^2 \cdot \sin^6(5\pi)$$

$$F_3(x) = \sin^6(5\pi(x^{0.75} - 0.05))$$

$$F_4(x) = \exp(-2\log(2)\cdot(\frac{x-0.1}{0.854})^2 \cdot \sin^6(5\pi(x^{0.75} - 0.05))$$

$$0 \le x \le 1$$

Table 2. Comparison of results on $F_1\ F_2\ F_3\ F_4$

Function	Num. Of global optima	Num. of evals (mean and std dev)	
		HPSO	SPSO
F_1	5	625 ± 195.12	1383 ± 242.95
F_2	1	399 ± 203.71	351 ± 202.35
F_3	5	653 ± 187.65	1248 ± 318.80
F_4	1	421 ± 221.493	503.33 ± 280.107

Experimental results on the above four functions are listed in Table 2 Results of SPSO are come from [5].

From the results we can see that, for these functions, our algorithm is more efficient than SPSO for finding multi global optima.

Table 3. Comparison of results on F_5 and F_6

Function	Num. of global optima	Num. of evals (mean and std dev)	
		HPSO	SCGA
F_5	2	840 ± 328.23	935
F_6	2	1380 ± 451.22	2811

Function F_5 and F_6 are come from [3], F_5 is called central peak function and F_6 is called five peak function. Experimental results on the two equations are listed in Table 3 Our algorithm can found all global optima in few iterations, one reason is that both the two global optima are located on the boundary of feasible region, in our algorithm, particles can explore the search space in large step, it is easy to reach the boundary.

$$F_5(x) = \begin{cases} 16c & for\ 0 \le c \le 10 \\ 32(15-c) & for\ 10 \le c \le 15 \\ 40(c-15) & for\ 15 \le c \le 20 \end{cases}$$

$$F_6(x) = \begin{cases} 80(2.5-c) & for\ 0 \le c \le 2.5 \\ 64(c-2.5) & for\ 2.5 \le c \le 5.0 \\ 64(7.5-c) & for\ 5.0 \le c \le 7.5 \\ 28(c-7.5) & for\ 7.5 \le c \le 12.5 \\ 28(17.5-c) & for\ 12.5 \le c \le 17.5 \\ 32(c-17.5) & for\ 17.5 \le c \le 22.5 \\ 32(27.5-c) & for\ 22.5 \le c \le 27.5 \\ 80(c-27.5) & for\ 27.5 \le c \le 30 \end{cases}$$

F_7, F_8 and F_9 are frequently used benchmark functions. F_7 Himmelblau's function has 4 global maxima solutions. F_8 six-hump camel back function has 2 global optima and F_9 Brannin RCOS function has 3 global optima. We compare our result on F_7 with the results reported in [5] by SPSO, the convergence speed is slower than SPSO, however, the SPSO algorithm cannot get 100% success rate on F_8 and F_9.

$$F_7(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$$
$$-6 \le x, y \le 6$$

$$F_8(X) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})\ x_1^2 + x_1x_2 + 4(x_2^2 - 1)x_2^2$$
$$-3 \le x_1 \le 3, -2 \le x_2 \le 2$$

$$F_9(X) = a(x_2 - bx_1^2 + cx_1 - d)^{\ 2} + e(1-f)\cos(x_1) + e$$
$$-5 \le x_1 \le 10, 0 \le x_2 \le 15,$$
$$a = 1, b = 5.1/(4\pi^2), c = 5/\pi,$$
$$d = 6, e = 10, f = 1/(8\pi)$$

Table 4. Comparison of results on F_7 F_8 and F_9

Function	Num. of global optima	Num. of evals (mean and std dev)	
		HPSO	SCGA
F_7	4	3792 ± 487.24	3155(SPSO)
F_8	2	2450 ± 452.24	1839
F_9	3	4849 ± 1149.37	8529

This is an n dimensional Shubert function. It has $n \cdot 3^n$ global optima which can be divided into 3^n groups. In two dimensional cases, it has 760 local minima, 18 of which are global minima with an objective function value of -186.73. These 18 global optima can be divided into 9 groups; each group has two optima close to each other. In three dimensional

cases, it has 81 global optima. We test our algorithm when $n=2$ and $n=3$ .Parameters and results are shown in Table 5.

Table 5. Comparison of results on Shubert Function

n	Parameters	Mean Evals	
		HPSO	SCGA
$n=2$	*PopSize*=1500, *SpecMin*=10, $\varepsilon=0.01$, *SpecMax*=25	57750	100000
$n=3$	*PopSize*=5000, *SpecMin*=30, $\varepsilon=0.01$,*SpecMax*=100	486200	850338

We test on each case 50 times, for $n=2$ we found all 18 global optima in 100% success rate, for $n=3$, we found all 81 global optima 48 times, 79 global optima were found in the other two tests within maximum 100000 iterations. Our experimental results show that HPSO is more efficient than SCGA for Shubert function on two and three dimensional case. Figure. (3) illustrate the pattern of evolution of the population during a typical HPSO run on two dimensional Shubert Function, black squares represent the best individuals found in that iteration. HPSO can locate the global cluster in 5 iterations and locate all 18 global optima in 10 iterations.

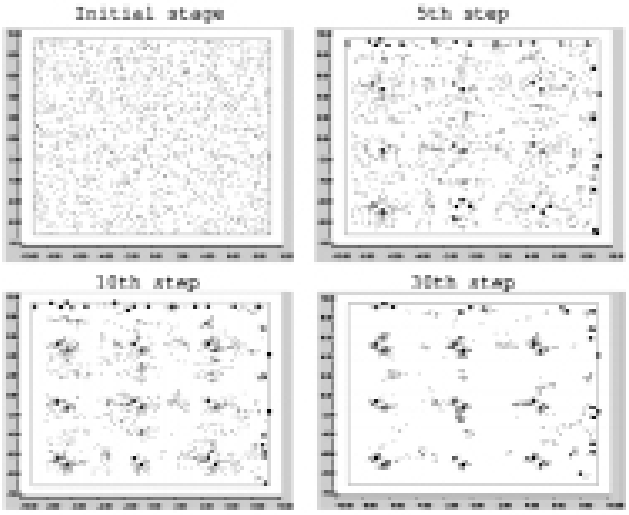

Figure. (3). Evolution of HPSO when applied to 2D Shubert function

B. High Dimensional Functions.

In this experiment, we used five standard benchmark function. Each function have one global optima with value equal to 0. When tested on these functions, we set PopSize=5000, MaxIteration=100,SpecMax=100, SpecMin=30. In Table 6, the data of algorithm PSO and KSwarm are from paper [16], while data of algorithm ExpVer and ES in Table 7 are from paper [17]. From the experiment results we can see that our method got an overall good performance and it is much better than basic PSO.

$$Sphere(x)=\sum_{i=1}^{d}x_i^2$$

$$DeJong(x)=\sum_{i=1}^{d}ix_i^4$$

$$Rastrigin(x)=\sum_{i=1}^{d}x_i^2+10-10\cos(2\pi x_i)$$

$$Griewank(x)=\frac{1}{4000}\sum_{i=1}^{d}x_i^2-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}})+1$$

Table 6. Experiment results on high dimensional functions

Function	Range	PSO	KSwarm
Sphere	(-50,50)	370.041	4.723
DeJong	(-20,20)	4346.714	4.609
Rosenbrock	(-100,100)	2.61e7	3.28e3
Rastrigin	(-5.12,5.12)	106.550	53.293
Griewank	(-600,600)	13.865	0.996

Table 7. Experiment results on high dimensional functions-continue

Function	Range	ExpVer	ES	HPSO
Sphere	(-50,50)	0	0	0.29
DeJong	(-20,20)	0	0	0.03
Rosenbrock	(-100,100)	47.75	50.8	29.15
Rastrigin	(-5.12,5.12)	63.22	57.19	13.34
Griewank	(-600,600)	0.038	0.002	0.31

5. Discussion

In this section, we will discuss the effect of parameters for performance of HPSO algorithm. Here, we use 2D and 3D Shubert function as examples, since they are the bench-

mark functions for multimodal optimization algorithms, they have many local optima and their global optima are unevenly distributed.

A. Effect of SpecMin and SpecMax.

Like other species evolution algorithms, such as SCGA and SPSO, species distance ra greatly affect the performance of algorithm. Although we introduced a mechanism to change ra adaptively during the running of algorithm, it's initial value should be properly set by Equation 2. In our algorithm, initial value of ra is decided by SpecMax, as we have explained in previous section, SpecMax represents the maximum number of high level groups, it is value should not be less than the number of global optima clusters. When tested on 2D Shubert function, SpecMax should not less than 9 and not less than 27 on 3D case. On 2D Shubert function we usually set SpecMax value from 15 to 30, if the value is bigger than

70, the algorithm can hardly found all global optima since too many groups will lead the algorithm converge to local optima.

Figure. (4) illustrates how ra changing with the iteration, from the Figureure we can see that ra will converge to a stable value in a few iterations. Here, we need to mention that ra represents the radius of high level groups, δ_{di} represents the minimum distance of global optima and its value is decreasing with iteration. Figure. (5) illustrate how groups numbers changing with the iteration, although high level group numbers relatively do not change with iteration, low level group numbers increasing since δ_{di} is decreasing.

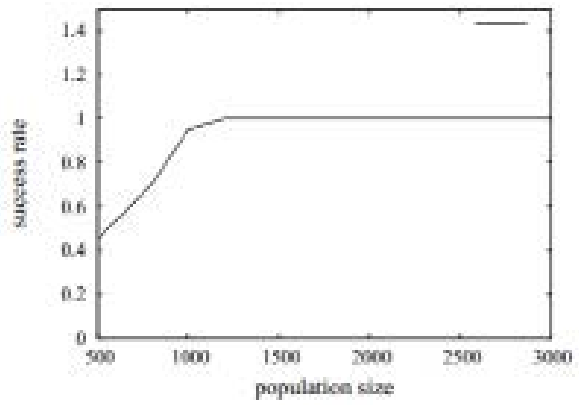

Figure 4. r_a changing with iteration

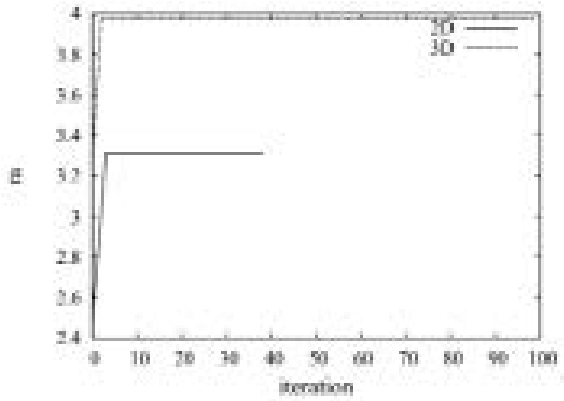

Figure 5. groups number changing with iteration

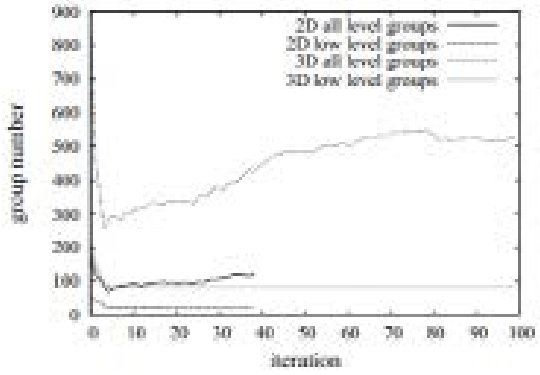

Figure 6. groups number changing with iteration

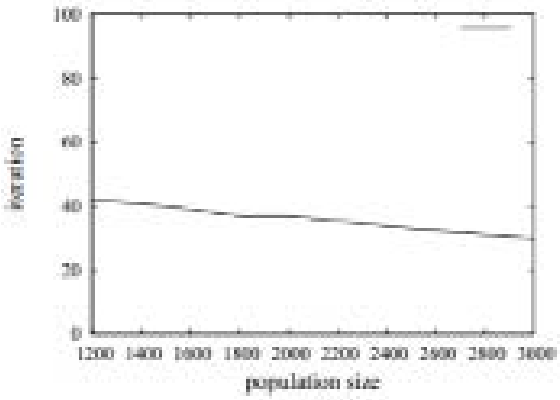

Figure 7. groups number changing with iteration

Conclusions

In this paper, we propose a new method to find all global minima of multimodal functions. Several previous researches indicated that PSO can locate the region of optimum faster than different evolution algorithm, but once in this region PSO progressed slowly, our experimental results demonstrated that PSO can also achieve a rapid convergence speed in the region of optimum. In addition, through partitioning searching space into several regions of different size, we improve the ability of particle swarm optimization to search at finer grain without losing its rapid convergence.

B. Effect of PopSize.

Population size of particles is another important parameter in our algorithm. Larger population size can increase the success rate of finding all global optima. When test on 2D Shubert function, the thresh value of population size is about 1200, as shown in Figure. (7). In 3D case, the value is about 5000. With the population size increasing, the iteration number slightly decreases. As shown in Figure. (6), when the population size doubled, the iteration number decreased 20%. Generally speaking, Large population size can improve the effective of algorithm while impair the efficiency.

Acknowledgements

This work was financially supported by the National Natural Science Foundation(61201246), and National Natural Science Foundation through the General Programs (51178368, 51478372).

References

1. Parsopoulos, Konstantinos E., and Michael N. Vrahatis. On the computation of all global minimizers through particle swarm optimization. Evolutionary Computation, IEEE Transactionson, 2004, pp. 211-224.

2. Pal, Narendra Singh, and Sanjeev Sharma. Robot PathPlaning using Swarm Intelligence: A Survey. METHODS, 2013, pp. 12-23.
3. Li, Jian-Ping, et al, A species conserving genetic algorithfor multimodal function optimization. Evolutionary computation, 2002, pp. 207-234.
4. Qu,Bo-Yang, Ponnuthurai Nagaratnam Suganthan and SwagatamDas, A distance-based locally informed particle swarm model for multimodal optimization. "Evolutionary Computation, IEEE Transactions on, 2013. pp. 387-402.
5. Li, Xiaodong, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. Genetic and Evolutionary Computation–GECCO 2004. Springer Berlin Heidelberg, 2013, pp. 169-177.
6. Peña, Jose Manuel, Jose Antonio Lozano, and Pedro Larrañaga, 2005.Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. Evolutionary Computation 13.1 : 43-66.
7. Liang, Yong, and Kwong-Sak Leung. Genetic Algorithm with adaptive elitist-population strategies for multimodal function optimization. Applied Soft Computing, 2011, pp. 2017-2034.
8. Ando, Shin, Jun Sakuma, and Shigenobu Kobayashi. Adaptive isolation model using data clustering for multimodal function optimization. Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM, 2005, pp. 301-308.
9. Li, Xiaodong. Niching without niching parameters: particleswarm optimization using a ring topology. Evolutionary Computation, IEEE Transactions on 14.1, 2010, pp. 150-169.
10. Blum, Christian, and Xiaodong Li. Swarm intelligence in optimization. Springer Berlin Heidelberg, 2008.
11. Phillips, Robert Brooks, et al. Physical biology of the cell. New York: Garland Science, 2009.
12. Gonçalves, José Fernando, and Mauricio GC Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers & Operations Research, 2009, pp. 179-190.
13. Eberhart, Russ C., and James Kennedy. A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science. Vol. 1. 1995.
14. Hu, Xiaohui, Yuhui Shi, and Russ Eberhart. Recent advances in particle swarm. IEEE congress on evolutionary computation. Vol. 1. Portland, 2004.
15. Kennedy, James, and Rui Mendes. Population structure and particle swarm performance, 2002.
16. Monson, Christopher K., and Kevin D. Seppi. The Kalman swarm. Genetic and Evolutionary Computation–GECCO 2004. Springer Berlin Heidelberg, 2004.
17. Clerc, Maurice, and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on 6.1, 2002. pp. 58-73.