PARALLEL CAT SWARM OPTIMIZATION

PEI-WEI TSAI[1], JENG-SHYANG PAN[1], SHYI-MING CHEN [2, 3], BIN-YIH LIAO[1], SZU-PING HAO[4]

[1] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan
[2] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
[3] Department of Computer Science and Information Engineering, Jinwen University of Science and Technology, Taipei County, Taiwan
[4] Department of Mechanical Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan
E-MAIL: pwtsai@bit.kuas.edu.tw, jspan@cc.kuas.edu.tw, smchen@mail.ntust.edu.tw, byliao@cc.kuas.edu.tw, lindaa@cc.kuas.edu.tw

Abstract:

We investigate a parallel structure of Cat Swarm Optimization (CSO) in this paper, and we call it Parallel Cat Swarm Optimization (PCSO). In the experiments, we compare Particle Swarm Optimization (PSO) with CSO and PCSO. The experimental results indicate that both CSO and PCSO perform well. Moreover, PCSO is an effective scheme to improve the convergent speed of Cat Swarm Optimization in case the population size is small and the whole iteration is less.

Keywords:

Evolutionary; swarm intelligent; computational intelligence; optimization; parallel swarm

1. Introduction

Computational Intelligence is a popular research topic in recent years. Many related algorithms were proposed, such as Cat Swarm Optimization (CSO) [1-2], Genetic Algorithm (GA) [5-8], Particle Swarm Optimization (PSO) [3, 9-10], and Ant Colony Optimization (ACO) [11-12]. These algorithms, except GA, are developed by simulating the intelligent behaviors of creatures based on swarm intelligence. In addition, they also have a common specific, i.e. they employ numbers of individuals for searching the optimum solutions to the benchmark.

In this paper, we propose a parallel scheme for CSO in order to advance its convergence and searching ability. The proposed algorithm is called Parallel Cat Swarm Optimization (PCSO). According to the experimental results, PCSO affords to search the optimum solutions with a few individuals and small amount of iterations than CSO and Particle Swarm Optimization (PSO).

2. Cat Swarm Optimization

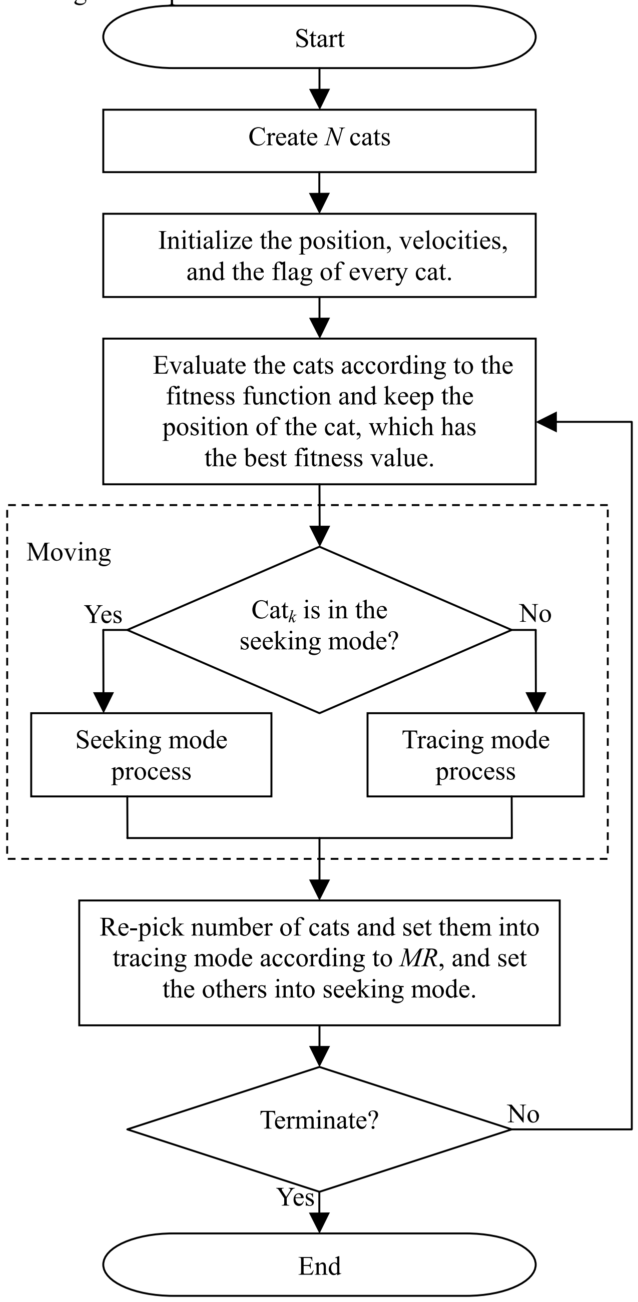
CSO is developed based on cats' behaviors. Applying CSO to solve problems of optimization, the first step is to decide how many individuals to use. The individuals in CSO is called cats, and every cat has its own position composed of *M* dimensions, velocities for each dimension, a fitness value representing the accommodation of the cat to the benchmark function, and a flag to identify whether the cat is in seeking mode or tracing mode. The final solution would be the best position of one of the cats in the solution space. CSO keeps the best solution until it reaches the end of the iterations. CSO has two sub modes, namely seeking mode and tracing mode. These two modes individually represent different procedures in the algorithm to imitate the behaviors of cats, and they are dictated to join with each other by a mixture ratio *MR*. The process of CSO is described as follows:

1. Create *N* cats in the process.
2. Randomly sprinkle the cats into the *M*-dimensional solution space and randomly assign values, which are in-range of the maximum velocity, to the velocities of every cat. Then haphazardly pick number of cats and set them into tracing mode according to *MR*, and the others set into seeking mode.
3. Evaluate the fitness value of each cat by applying the positions of cats into the fitness function, which represents the criteria of our goal, and keep the best cat into the memory. Note that we only need to remember the position of the best cat (x_{best}) due to it represents the best solution so far.
4. Move the cats according to their flags, if cat$_k$ is in the seeking mode, apply the cat to the seeking mode process, otherwise apply it to the tracing mode process.

The process steps are presented in 2.1 and 2.2.

5. Re-pick number of cats and set them into tracing mode according to *MR*, then set the other cats into seeking mode.
6. If the termination condition is satisfied, terminate the program, otherwise repeat Step3 to Step5.

Figure 1 represents the flowchart of CSO.



Start
Create *N* cats
Initialize the position, velocities, and the flag of every cat.
Evaluate the cats according to the fitness function and keep the position of the cat, which has the best fitness value.
Moving
Yes
Cat$_k$ is in the seeking mode?
No
Seeking mode process
Tracing mode process
Re-pick number of cats and set them into tracing mode according to *MR*, and set the others into seeking mode.
Terminate?
No
Yes
End

Figure 1. The flowchart of CSO

2.1. Seeking Mode

In the process of CSO, seeking mode is presented to model the cat during a period of resting but being alertly looking around the environment for its next movement. Four essential factors are applied in seeking mode, namely, seeking memory pool (*SMP*), seeking range of the selected dimension (*SRD*), counts of dimension to change (*CDC*) and self position consideration (*SPC*). *SMP* is used to define the size of seeking memory of each cat, which indicates any point sort by the cat. The cat would pick a point from the memory pool, according to rules described later. *SRD* declares the mutative ration for the selected dimensions. If a dimension is selected for mutation, the difference between the old and new values may not be out of range, the range is defined by *SRD*. *CDC* tells how many of the dimensions will be varied. All these factors play important roles in seeking mode. *SPC* is a Boolean valued variable, and indicates whether the point at which the cat is already standing will be one of the candidate points to move to. SPC cannot influence the value of SMP. Seeking mode is described as follows:

1. Make *j* copies of the present position of cat$_k$, where $j = SMP$. If the value of *SPC* is true, let $j = (SMP - 1)$, then retain the present position as one of the candidates.
2. For each copy, according to *CDC*, randomly plus or minus *SRD* percents the present values and replace the old ones.
3. Calculate the fitness values (*FS*) of all candidate points.
4. If all *FS* are not exactly equal, calculate the selecting probability of each candidate point according to equation (1); otherwise set all the selecting probability of each candidate point to be *1*.
5. Randomly pick the point to move to from the candidate points, and replace the position of cat$_k$.

$$P_i = \frac{|FS_i - FS_b|}{FS_{\max} - FS_{\min}}, \text{ where } 0 < i < j \qquad (1)$$

If the goal of the fitness function is to find the minimum solution, let $FS_b = FS_{\max}$, otherwise $FS_b = FS_{\min}$.

2.2. Tracing Mode

Tracing mode models the case of the cat in tracing targets. Once a cat goes into tracing mode, it moves according to its own velocities for each dimension. The action of tracing mode can be described as follows:

1. Update the velocities for every dimension $v_{k,d}(t)$ for the cat$_k$ at the current iteration according to equation (2).
2. Check if the velocities are in the range of maximum velocity. In case the new velocity is over-range, it is set equal to the limit.
3. Update the position of cat$_k$ according to equation (3).

$$v_{k,d}(t)=v_{k,d}(t-1)+r_1\cdot c_1\cdot\left[x_{best,d}(t-1)-x_{k,d}(t-1)\right],\quad d=1,2,\ldots,M \tag{2}$$

where $x_{best,d}(t-1)$ is the position of the cat, who has the best fitness value, at the previous iteration; $x_{k,d}(t-1)$ is the position of cat$_k$ at the previous iteration, c_1 is a constant and r_1 is a random value in the range of [0, 1].

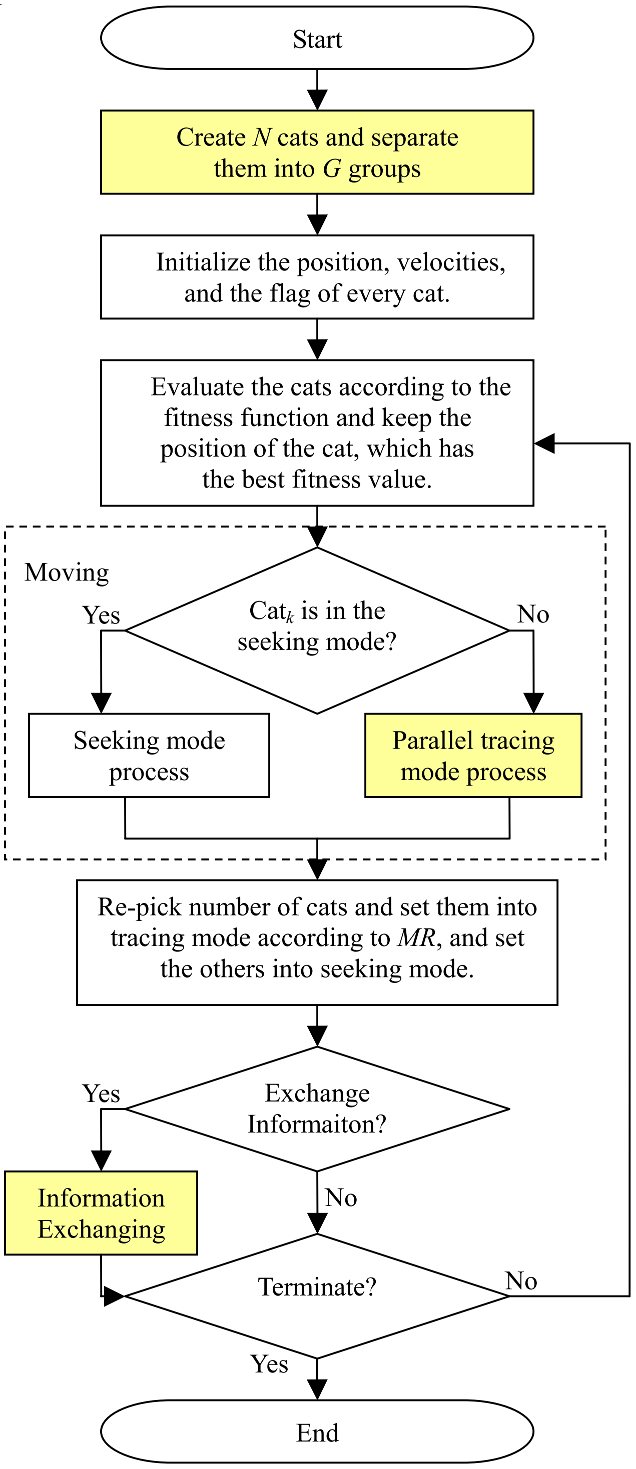
$$x_{k,d}(t)=x_{k,d}(t-1)+v_{k,d}(t) \tag{3}$$

3. Parallel Cat Swarm Optimization

The precedents of splitting the amount of the population into several sub-populations to construct the parallel structure can be found in several algorithms, such as Island-model Genetic Algorithm or Parallel Genetic Algorithm [4] and Parallel Particle Swarm Optimization Algorithm with Communication Strategies [3]. Each of the sub-populations evolves independently and shares the information they have occasionally. Although it results in the reducing of the population size for each sub-population, the benefit of cooperation is achieved.

By inspecting the structure of CSO, the individuals work independently when they stay in the seeking mode. Oppositely, they share the identical information, the global best solution, as they know according to their knowledge to the present in the tracing mode. In PCSO, we still follow the structure of this framework.

Underneath the structure of CSO, the individuals, most of the time, works as a stand along system. To parallelize these individuals in CSO, we modify the procedure in tracing mode to make it seems more cooperative. In PCSO, we first separate the individuals into several sub-populations. Hence, the individuals in tracing mode do not move forward to the global best solution directly, but they move forward to the local best solution of its own group in general. Only if the programmed iteration achieved, the sub-populations pop the local best solutions at present and randomly pick a sub-population to replace the worst individual in the selected sub-population. The procedure of PCSO is described as follows:


Start
Create N cats and separate them into G groups
Initialize the position, velocities, and the flag of every cat.
Evaluate the cats according to the fitness function and keep the position of the cat, which has the best fitness value.
Moving
Yes
Cat$_k$ is in the seeking mode?
No
Seeking mode process
Parallel tracing mode process
Re-pick number of cats and set them into tracing mode according to MR, and set the others into seeking mode.
Yes
Exchange Informaiton?
Information Exchanging
No
Terminate?
No
Yes
End
Figure 2. The Flowchart of PCSO

Basically, the main framework of PCSO is similar to CSO. At the beginning of the algorithm, *N* individuals are created and separate into *G* groups. The Parallel tracing mode process and the information exchanging process are described in 3.1 and 3.2. Assume that we set *G* equals to one, and then the PCSO returns into the original CSO.

3.1. Parallel Tracing Mode Process

The parallel tracing mode process can be described as follows:

1. Update the velocities for every dimension $v_{k,d}(t)$ for the cat$_k$ at the current iteration according to equation (4).
2. Check if the velocities are in the range of maximum velocity. In case the new velocity is over-range, it is set equal to the limit.
3. Update the position of cat$_k$ according to equation (5).

$$v_{k,d}(t) = v_{k,d}(t-1) + r_1 \cdot c_1 \cdot \left[x_{l_{best},d}(t-1) - x_{k,d}(t-1)\right], \quad d = 1,2,\ldots,M \tag{4}$$

where $x_{l_{best},d}(t-1)$ is the position of the cat, who has the best fitness value, at the previous iteration in the group that cat$_k$ belongs to.

$$x_{k,d}(t) = x_{k,d}(t-1) + v_{k,d}(t) \tag{5}$$

3.2. Information Exchanging

The procedure forces the sub-populations exchange their information, and achieves somehow the cooperation. We define a parameter *ECH* to control the exchanging of the information between sub-populations. The information exchanging is applied once per *ECH* iterations.

The information exchanging consists of three steps. Firstly, we pick up a group of the sub-populations sequentially and sort the individuals in this group according to their fitness value; secondly, we randomly select a local best solution from an unrepeatable group; thirdly, the individual, whose fitness value is worst in the group, is replaced by the selected local best solution.

4. Experiments

In this section, we evaluate PCSO by three well-known test functions and compare it with CSO and PSO with weighting factor (PSO with WF). In addition, we notice that PCSO performs convergence faster than CSO and PSO with weighting factor. The details are described as follows.

4.1. Test Functions

We apply PCSO, CSO and PSO with weighting factor into three test functions to compare the performance. These test functions, namely, Rosenbrock function, Rastrigrin function, and Griewank function, are listed as follows:

$$f_1(x) = \sum_{d=2}^{M}\left[100 \times (x_d - x_{d-1})^2 + (x_{d-1} - 1)^2\right] \tag{6}$$

$$f_2(x) = \sum_{d=1}^{M}\left[x_d^2 - 10 \cdot \cos(2\pi x_d)^2 + 10\right] \tag{7}$$

$$f_3(x) = \frac{1}{400}\sum_{d=1}^{M} x_d^2 - \prod_{d=1}^{M}\cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \tag{8}$$

In the experiments for all test function, we aim at finding the minimum of the fitness value, in other words, the goal of our experiments is to minimize the fitness function. For each test function, we limit the initial ranges for every dimension according to Table 1.

Table 1. The limintation ranges for the test functions

Function Name	Limintation Range
Rosenbrock	$x_d \in [15, 30]$
Rastrigrin	$x_d \in [2.56, 5.12]$
Griewank	$x_d \in [300, 600]$

Note that the limits are only suitable for the initialization, when the process moves to the evaluation and movement stage, there is no limit on any dimension for every individual.

4.2. Parameter Settings

The parameter settings for both PCSO and CSO are listed in Table 2. Moreover, for PCSO, the group number *G* is set to *2* and *4* as two individual experiments. And the *ECH* is set to *20*.

The parameter setting for PSO with WF is listed in Table 3. The same conditions for all these algorithms are initial ranges, which have been shown in Table 1.

During the experiments, we focus on the case when the optimization has to be done with only a small size of the population and less iterations. Hence, the dimensions of all fitness functions are set to be *30*, the population sizes are all be set to *16*. For each algorithm, we apply *500* iterations per cycle and *50* cycles per test function, and the final results are composed of the average of the *50* cycles. Besides, the *SPC* flags for PCSO and CSO are set to be true, and for all the experiments, the maximum velocities for each test function are shown in Table 4. The usage of the parameters for PSO with WF can be found in [9].

Table 2. Parameter Settings for PCSO and CSO

Parameter	Value or Range
MP	*5*
SRD	*20%*
CDC	*80%*
MR	*2%*
c_1	*2.0*
r_1	*[0, 1]*

Table 3. Parameter Settings for PSO with WF

Parameter	Value or Range
Initial Weight	*0.9*
Final Weight	*0.4*
c_1	*2.0*
r_1	*[0, 1]*
c_2	*2.0*
r_2	*[0, 1]*

Table 4. Maximum velocities

Function Name	Maximum Velocity
Rosenbrock	100.0
Rastrigrin	10.0
Griewank	600.0

4.3. Experimental Results

The experimental results are presented in Figure 3 to Figure 5. The horizontal axis represents the iterations, and the vertical axis represents the fitness.

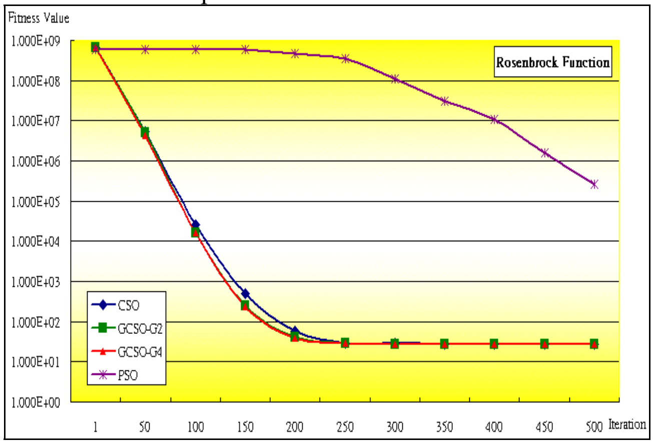


Figure 3. The experimental result of Rosenbrock Function

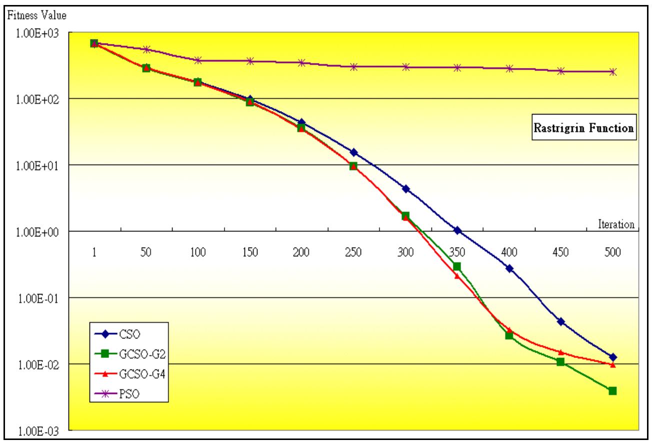


Figure 4. The experimental result of Rastrigrin function

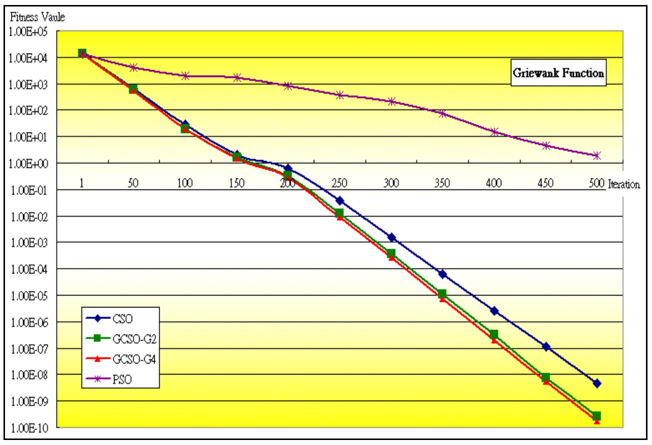


Figure 5. The experimental result of Griewank function

4.4. Discussions and Conclusions

The experimental results indicate that PCSO performs better than CSO and much better than PSO when the population size is small and the iteration is less. Moreover, we would like to inspect the convergence between these algorithms. The last *20* iterations of these test functions are stretched and displayed in Figure 6 to Figure 8. Due to the outcomes from PSO is much larger than PCSO and CSO, the results of PSO in Figure 6 and Figure 7 are not presented.

The results exhibit that the convergence and the searching ability of PCSO is higher in case the iteration is less and the population size is small. The information exchanging plays a key role improving the searching ability of CSO, and it performs the cooperation between sub-populations. In future works, we plan to focus on the seeking mode in PCSO, and lead the cooperation into it.

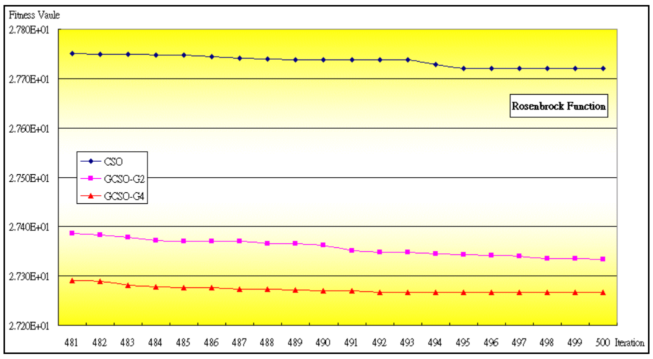


Figure 6. The stretched result of Rosenbrock function

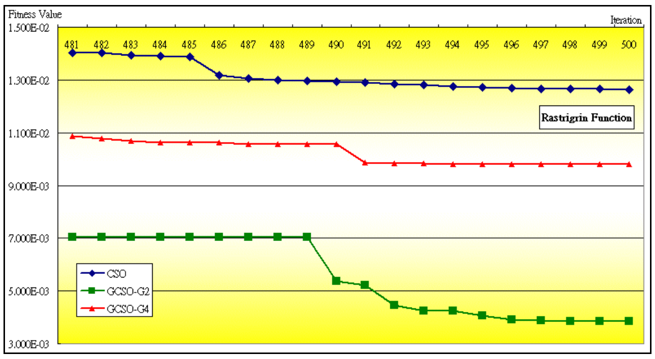


Figure 7. The stretched result of Rastrigrin function

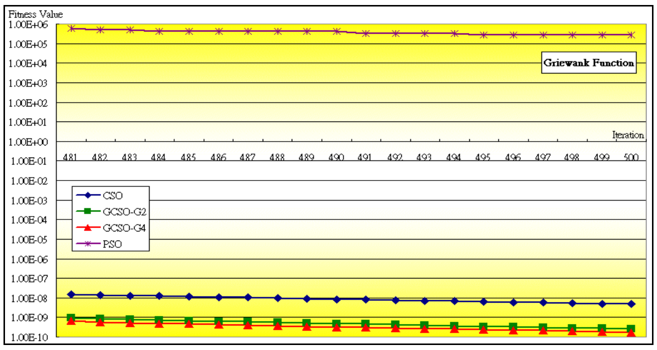


Figure 8. The stretched result of Griewank function

References

[1] Shu-Chuan Chu and Pei-Wei Tsai, “Computational Intelligence based on the Behavior of Cats”, International Journal of Innovative Computing, Information and Control, vol. 3, no. 1, pp. 163-173, Feb. 2007.

[2] Shu-Chuan Chu, Pei-Wei Tsai and Jeng-Shyang Pan, “Cat Swarm Optimization”, Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence LNAI 4099, Guilin, pp. 854-858, Aug. 2006.

[3] Jui-Fang Chang, Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, “A Parallel Particle Swarm Optimization Algorithm with Communication Strategies”, Journal of Information Science and Engineering, vol.21, no.4, pp.809-818, 2005.

[4] D. Abramson, and J. Abela, “A Parallel Genetic Algorithm for Solving the School Timetabling Problem”, Technical Report, Division of Information Technology, CSIRO, 1991.

[5] D. E. Goldberg, “Genetic Algorithm in Search. Optimization and Machine Learning”, Addison-Wesley Publishing Company, 1989.

[6] J. S. Pan, F. R. Mclnnes, and M. A. Jack, “Application of Parallel Genetic Algorithm and Property of Multiple Global Optima to VQ Codevector Index Assignment”, Electronics Letters, vol.32, no.4, pp.296-297, 1996.

[7] KwanWoo Kim, Mitsuo Gen, and MyoungHun Kim, “Adaptive Genetic Algorithms for Multi-resource Constrained Project Scheduling Problem with Multiple Modes”, International Journal of Innovative Computing, Information and Control, vol.2, no.1, pp.41-49, 2006.

[8] Yoichiro Maeda, and Qiang Li, “Parallel Genetic Algorithm with Adaptive Genetic Parameters Tuned by Fuzzy Reasoning”, International Journal of Innovative Computing, Information and Control, vol.1, no.1, pp.95-107, 2005.

[9] Y. Shi, and R. Eberhart, “Empirical study of particle swarm optimization”, Congress on Evolutionary Computation, pp.1945-1950, 1999.

[10] Nobuhiro Iwasaki and Keiichiro Yasuda, “Adaptive Particle Swarm Optimization Using Velocity Feedback”, International Journal of Innovative Computing, Information and Control, vol.1, no.3, 2005.

[11] M. Dorigo, and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem”, IEEE Trans. on Evolutionary Computation, vol.26, no.1, pp.53-66, 1997.

[12] S.-C. Chu, J. F. Roddick, and J. S. Pan, “Ant colony system with communication strategies”, Information Sciences, vol.167, pp.63-76, 2004.