2008년도 대한전기학회 EMECS학회 추계학술대회 논문집 (2008.10.23-10.25)

Utilizing Particle Swarm Optimization into Multimodal Function Optimization

Minh-Trien Pham, Nyambayar Baatar, 고창섭
충북대학교 전기공학과

Utilizing Particle Swarm Optimization into Multimodal Function Optimization

Minh-Trien Pham, Nyambayar Baatar, Chang Seop Koh
School of Electrical & Computer Engineering, Chungbuk National University

Abstract - There are some modified methods such as K-means Clustering Particle Swarm Optimization and Niching Particle Swarm Optimization based on PSO which aim to locate all optima in multimodal functions. K-means Clustering Particle Optimization could locate all optima of functions with finite number of optima. Niching Particle Swarm Optimization is able to locate all of optima but high computing time. Because of those disadvantages, we proposed a new method that could locate all of optima with reasonal time. We applied our method and others as well to analytic functions. By comparing the outcomes, it is shown that our method is significantly more effective than the two others.

1. Introduction

The Particle Swarm Optimization (PSO) algorithm was proposed by James Kennedy and Russell Eberhart in 1995 [1]. It is known as the method which has more advantages than Genetic Algorithm and Evolution Strategy in unimodal optimization. One of the most advantages is fast converged speed. That is reason why nowadays there are many methods based on PSO to locate multi solution of multimodal functions.

K-means Clustering PSO (K-PSO) use standard K-means clustering method to divide initial particles into K groups. Each group will search local optimum in sphere activity of the group. So, K-PSO could locate maximum K local minima [2].

Niching Particle Swarm Optimization (NPSO) is combined Niching algorithm and PSO. Each initial particle searches local optimum until it makes sure that local optimum existence. Once the local existence of a local optimum is found, the particle and its nearest one create a group to exactly locate the local optima [4]. By this method, it is able to locate all local optima but the converged speed is slow.

This paper briefly reviews the methods based on PSO. And then we proposed a new method named Couple Particle Swarm Optimization to overcome the disadvantages of K-PSO and NPSO.

2. Background

2.1 K-means Clustering Particle Optimization

This method is accomplished by applying a standard clustering algorithm and PSO. In this way each cluster of particles tends to perform a local search in the function domain and to locate different optima [2].

Clustering is performed after the swarm initialization and then repeated a fixed number of times during the swarm simulation. Between two clustering applications the swarm or more precisely, the sub swarms corresponding to the various clusters follows their normal dynamics. In fact, the multiple applications of the clustering algorithm are meant keep track of the swarm dynamics: particles in different clusters at early stages of the simulation can end up in the same cluster as they move towards the same local optimum or, at the contrary, a single cluster can be split in two as some of its particles fly towards of different optimum [2].

There are many kind of clustering, in this method using the standard k-means algorithm to perform the clustering of the particles, but with two adjustments.

1. For single clustering procedure the k-means algorithm is repeated r_k times in order to reduce the variance due to random centroid initialization. The clustering with minimum sum-of-squared-error is then chosen

2. After the clusters have been formed, the ones with a number of particles higher than the average are reduced. The exceeding particles are randomly removed from them and reinitialized.

2.2 Niching Particle Swarm Optimization

NPSO was known as the method which can successfully locate all maxima on a small set of test functions during all simulation runs [4].

In NPSO, PSO was modified by the Guaranteed Convergence Particle Swarm Optimization (GCPSO) algorithm [6]. This optimization was created because of reason is when the particle's position equal to particle's local position and equal to group best position, the velocity update only depends solely on previous velocity term. When a particle approaches the global best solution, its velocity property approaches zero, which implies that eventually all particles will stop moving. This behavior does not guarantee convergence to a global best solution, or event a local best, only to a best position found thus far [4].

n particles was generated, they was called main swarm particles. At the first time, main swarm particles were trained using cognition only model. This arrangement allows each particle to perform a local search. After that, main swarm particles were updated fitness, if a particle's fitness showed very little change over a small number of iterations of the learning algorithm, a sub swarm is created with the particle and its closest topology logical neighbor. For each sub swarm, sub swarm particles were trained using once iteration of the GCPSO algorithm. After sub swarm particles update fitness, their swarm radius also was updated by calculate the maximum distance from sub swarm best particle with the furthest particle in that sub swarm. During moving, if any main warm particle located inside the sub swarm which absorb that main swam particle, this main swarm particle will become the sub swarm particle. So at that time, number of main swarm particle will be reduced and number particle in that sub swarm will be increased. When sub swarms moving, in some case they will tend toward the same local optimum. In that case, they will be merged and became one sub swarm. This process was repeated until stopping conditions are met.

2.3 Couple Particle Swarm Optimization

In this section, the couple PSO algorithm can locate all optima for multimodal function is discussed. There are n main particles were assigned uniformly. All main particles aim to define the direction which has local optima. After that a couple of particles will locate the local optimum exactly. The scheme of couple PSO algorithm is shown in Figure 1.

After main particles are initialized, they will search their local area by using PSO. In conventional PSO, the velocity was combine by two modal, they were named cognition only modal and social only modal. In the equation (1), they are second and third item respectively.

In PSO, with n main particles are generated in search space of d-dimensions. For each particle i its velocity vector $\mathbf{v}_i$ is updated at iteration t according to the equation:

```
1. Initialize main particles
2. Main particles local searching
3. Update fitness of each main swarm particle.
4. Create new particle couple.
    5. For each couple:
    5.1 Move using conventional PSO.
    5.2 Update each particle's fitness.
    5.3 Update couple radius
6. If possible, eliminate main particles and couples
7. Repeat from 2 until stopping conditions are met.
```

Figure 1

$$\mathbf{v}_i(t+1) = \omega\cdot\mathbf{v}_i(t) + c_1\cdot r_1\cdot(\mathbf{p}_i(t)-\mathbf{x}_i(t)) + c_2\cdot r_1\cdot(\mathbf{g}(t)-\mathbf{x}_i(t)) \quad (1)$$

In this model, velocity of particle only depends on its own experience as the equation (2).

$$\mathbf{v}_i(t+1) = \omega\cdot\mathbf{v}_i(t) + c_1\cdot r_1\cdot(\mathbf{p}_i(t)-\mathbf{x}_i(t)) \quad (2)$$

where $\mathbf{x}_i$, $\mathbf{v}_i$, $\mathbf{p}_i$ describe position, velocity and particle best of particle's i-th respectively.

Reason why in this step cognition only modal was used but not conventional PSO or social only modal because of locating local optimum ability of this modal [3]. And the position was calculated by:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1); i = 1,\cdots,n \quad (3)$$

After moved, main particles will be updated fitness. In minimize problem, if the function value of particle's position at the current iteration is better than previous one. The particle best position of the next iteration will be that particle's position.

$$\begin{aligned}\mathbf{p}_i(t+1) &= \mathbf{x}_i(t+1) \quad \text{if } f(\mathbf{x}_i(t+1)) > f(\mathbf{p}_i(t)) \\ &= \mathbf{p}_i(t) \quad \text{if } f(\mathbf{x}_i(t+1)) \le f(\mathbf{p}_i(t))\end{aligned} \quad (4)$$

where $f(\mathbf{p}_i(t))$ is the function value of particle best i-th at iteration t.

To create a new couple, a prediction method was used. We can easy recognize that if the particle tends toward local optimum the function value of this particle's position is smaller than previous. So, if any particle tends toward local optimum, a new couple is created by generated a new particle beside its self, both of them were combined a couple. With two particles, it is enough to apply conventional PSO.

Equation (5) shows the velocity update of each particle in couple k-th.

$$\mathbf{v}_j^k(t+1) = \omega\cdot\mathbf{v}_j^k(t) + c_1\cdot r_1\cdot(\mathbf{p}_j^k(t)-\mathbf{x}_j^k(t)) + c_2\cdot r_2\cdot(\mathbf{c}^k(t)-\mathbf{x}_j^k(t)) \quad (5)$$

and each particle updates position:

$$\mathbf{x}_j^k(t+1) = \mathbf{x}_j^k(t) + \mathbf{v}_j^k(t+1) \quad (6)$$

where $\mathbf{x}_j^k(t)$, $\mathbf{v}_j^k(t)$, $\mathbf{p}_j^k(t)$ are position, velocity and particle best of j-th particle in couple k-th at iteration t.

And the best position in couple k-th at iteration t is $\mathbf{c}^k(t)$. The particle best and couple best are updated at next step.

$$\mathbf{p}_j^k(t+1)=\mathbf{x}_j^k(t+1) \text{ if } f(\mathbf{x}_j^k(t+1)) > f(\mathbf{p}_j^k(t)) \\ =\mathbf{x}_j^k(t) \text{ if } f(\mathbf{x}_j^k(t+1)) \le f(\mathbf{p}_j^k(t)) \quad (7)$$

And the couple's best will be the better particle's position in couple:

$$\mathbf{c}^k(t+1)=\arg\min_j f(\mathbf{p}_j^k(t+1)); j=1,2 \quad (8)$$

Update couple radius:

$$R^k=\text{distance}(x_1^k,x_2^k);\text{distance}(x_1,x_2)=\|x_1-x_2\| \quad (9)$$

where R^k is couple radius of k-th couple. Because there are two particles in couple, so radius will be distance between them and the center of k-th couple is $\mathbf{c}^k$.

Eliminate particles:

$$\text{distance}(\mathbf{c}^k,\mathbf{x}_i)<R^k \quad (10)$$

If any main particle moves inside couple area, or distance between couple best $\mathbf{c}^k$ and main particle $\mathbf{x}_i$ is smaller than radius of k-th couple (10), this main particle should be eliminated. Because if a new couple is created by this main particle which will locate the same local optimum with the k-th couple.

Eliminate couples:

$$\text{distance}(\mathbf{c}^k,\mathbf{c}^m)<(R^k+R^m) \quad (11)$$

If the distance between of any two couple best is smaller than total radius of them as equation (11), the couple has worse function value will be eliminated. Eliminate method is applied to reduce number of function call, that means it reduces computation time.

Stopping conditions: There are two stop conditions, Maximum iteration equal 10,000 and all couple radii smaller than epsilon after 10 iterations. Actually, epsilon represents how accuracy of results. In this paper, epsilon equal 10^{-6} was chosen. If all of them are reached, the program will stop.

3. Experimental Results

In this section, three methods were analyzed in 2D design space. Our target is to locate all minima of the following functions:

$$F_1(\mathbf{x})=0.01\sum_{i=1}^{2}\left((x_i+0.5)^4-30x_i^2-20x_i\right); \\ \mathbf{x}\in[-5.12;5.12]^2;4\,\text{optima} \quad (12)$$

$$F_2(\mathbf{x})=(x_1^2+x_2-11)+(x_1+x_2^2-7)^2-200 \\ \mathbf{x}\in[-5;5]^2;4\,\text{optima} \quad (13)$$

$$F_3(\mathbf{x})=\sum_{i=1}^{N}\left\{x_i^2-10\cos(2\pi x_i)+10\right\} \\ \mathbf{x}\in[-1.3;1.3]^N;9\,\text{optima} \quad (14)$$

$$F_4(\mathbf{x})=\sum_{i=1}^{N}x_i\sin(\sqrt{x_i});\mathbf{x}\in[-80;60]^N;9\,\text{optima} \quad (15)$$

$$F_5(\mathbf{x})=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \\ \mathbf{x}\in[-9.5;9.5]^N;17\,\text{optima} \quad (16)$$

With all method, number of maximum iteration in stopping condition was set 10,000 and number of initial particles is 70. Those initial particles are generated by using Latin Hyper Design method and are used by all three methods in all trials.

Comparison performance of three methods are summarized in Table I, the percentage performance is calculated by ratio number of successfully locate all optima per number of trials.

The results show that NPSO and CPSO could locate all optima with all trials. K-PSO is performance little worst than the others.

The reason why CPSO could locate all local optima with high performance is each particle has chance locating its own local optimum.

Now we will concentrate to compare NPSO and CPSO, table II showed average number of iteration of all trials when these algorithms converged. It is shown that converged speed of CPSO is faster than NPSO. The reason is NPSO using cognition only modal until found local optimum while CPSO using conventional PSO to found and locate exactly local optimum. By the way, the converged speed of conventional PSO is faster than cognition only modal is proven by Kennedy [3].

Table III showed average number of function call. Number of function call is the most importance parameter in optimization problem. It is especially in real optimization problem. It concerns Finite Element Method calculation. Table III showed that average number of function call in CPSO is much smaller than NPSO. Even number of iteration almost the same, average number of function call in CPSO is still smaller than NPSO. The first reason is CPSO only has two particles in each couple. And the second, the number of couple is reduced while moving if they move toward the same local optimum.

4. Conclusion

In this paper, CPSO was compared with K-PSO and NPSO. As the results, CPSO could locate all local optima as well as NPSO and performance better than K-PSO in complicated analysis functions. Furthermore, when comparing converged speed and number of function call, it is shown that converged speed of CPSO is faster than NPSO. And the most importance parameter is number of function call table III that CPSO used smaller number of calling function with NPSO. It is mean that CPSO save more computing time than NPSO. These results had importance signification in real optimization problems.

Future research will apply CPSO into higher dimension multimodal functions and real multimodal optimization problems.

5. References

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc. IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942-1948.

[2] Alessandro Passaro and Antonina Starita, "Clustering particles for multimodal function optimization", Giorata di studio Italiana Sul Calcolo Eloluzionistico, 2006.

[3] J. Kennedy, "The particle swarm: social adaptation of knowledge," in Proceedings of IEEE Congress on Evolutionary Computation (CEC '97), pp. 303308, Indianapolis, Ind, USA, April 1997.

[4] Brits, R., Engelbrecht, A. P., van den Bergh, F.: A Niching Particle Swarm Optimizer. Conference on Simulated Evolution and Learning, Singapore (2002)

[5] Minh-Trien Pham, Nyambayar Baatar, Chang Seop Koh, "Couple Particle Swarm Optimization for Multimodal Functions", Proceedings of the KIEE IMECS Annual Spring Conference 2008, p 44-46, 2008.

[6] F van den Bergh, An Analysis of Particle Swarm Optimizers, PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

TABLE I
% experiments locating all optima

Function	K-PSO	NPSO	CPSO
$F_1(\mathbf{x})$	100%	100%	100%
$F_2(\mathbf{x})$	100%	100%	100%
$F_3(\mathbf{x})$	83%	100%	100%
$F_4(\mathbf{x})$	0%	100%	100%
$F_5(\mathbf{x})$	14%	100%	100%

TABLE II
Average number of iteration when algorithm was converged

Function	NPSO	CPSO
$F_1(\mathbf{x})$	1,833.57	750.45
$F_2(\mathbf{x})$	1,647.88	770.79
$F_3(\mathbf{x})$	2,983.34	954.28
$F_4(\mathbf{x})$	2,862.88	1,188.37
$F_5(\mathbf{x})$	3,224.49	1,096.34

TABLE III
Average number of function call when algorithm was converged

Function	NPSO	CPSO
$F_1(\mathbf{x})$	128,349.90	25,812.24
$F_2(\mathbf{x})$	115,351.60	25,435.78
$F_3(\mathbf{x})$	208,833.80	35,763.16
$F_4(\mathbf{x})$	200,401.60	45,245.28
$F_5(\mathbf{x})$	225,714.30	58,673.34

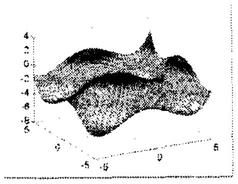
Figure 2. Function $F_1(\mathbf{x})$

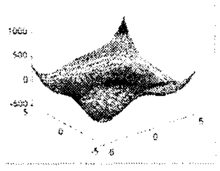
Figure 3. Function $F_2(\mathbf{x})$

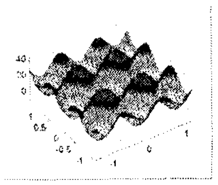
Figure 4. Function $F_3(\mathbf{x})$

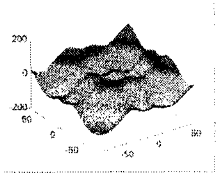
Figure 5. Function $F_4(\mathbf{x})$

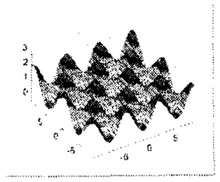
Figure 6. Function $F_5(\mathbf{x})$