Mutation Period Calculation for Particle Swarm Optimization

C. Ratanavilisagul,B. Kruatrachue, "Mutation Period Calculation for Particle Swarm Optimization," in Proceedings of the International Symposium on Technology for Sustainability (ISTS), 2011.

Chiabwoot Ratanavilisagul[1], Boontee Kruatrachue[1]

[1]Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand 10520
Department of Computer Engineering
Email: chaibwoot@hotmail.com, booontee@yahoo.com

Abstract— **Particle Swarm Optimization (PSO) algorithm use particles to search for better solution in solution space by changing particles position toward the best particles. One problem of PSO is trapping in local optima. Many researchers use mutation operation in order to avoid the trapping. This paper investigates when to apply the mutation in PSO algorithm. The iteration period to applied mutation depends on the average number of local optimum occurred in PSO. By applied normal PSO to the problem for some limited number of iteration, this average iteration period between each local optima can be observed and can be used in a normal run. By comparing the results on a set of benchmark test functions, the results show that the PSO mutation with the proposed iteration period yields better results.**

***Index Terms*— swarm intelligence, Particle swarm optimization, Cauchy mutation.**

I. Introduction

Kennedy and Eberhart [1,2] introduced Particle Swarm Optimization (PSO) in 1995. It is motivated from the behavior of flying bird and their communication mechanism in solving optimization problems. PSO has many advantages such as rapid convergence, simplicity, and little parameters to be adjusted. Its main disadvantage is trapping in local optimum and premature convergence.

To avoid local optima and premature convergence, many researchers [3-8] increased searching diversity of PSO by mutation operator.

This paper investigates when to applied mutation operation in PSO. Normally, mutation should occur when PSO trap in local optima. Researchers suggested applied with in some iteration period based on mutation rate [7] and mutation based on particles diversity [8].

The result showed that mutation operators for PSO can increase optimization performance and choosing mutation operators dependent on the problem.

In this paper, the mutation period is obtained by applying normal PSO to the problem of interest for a specific time and observed the average iteration period that PSO trap in local optima. Then this period is used for PSO to apply mutation in a longer run.

A set of benchmark test functions is used to compare the PSO with the proposed mutation period and the mutation from diversity and mutation rate. The results show that the proposed mutation period yield the best results in all test functions.

The rest of this paper is organized as follows. Section 2 explains basic PSO, Mutation Operators, PSO with mutation and when to apply mutation operator. Section 3 explains mutation period calculation for PSO. Section 4 explains the Benchmark Functions, describes the experiment setup and presents the experiment results. Section 5 concludes the paper with a brief summary.

II. Related Work

A. Particle Swarm Optimization

In the standard PSO, Each particle in PSO shares the information with neighbor particle. Particles search space by following the previous global best positions and the previous best positions of itself.

The PSO algorithm starts with random particles position and velocity. Each iteration, particle updates its position and velocity as follows:

$$X_{id}^{'} = X_{id} + V_{id}^{'} \quad (1)$$

$$V_{id}' = \omega V_{id} + \eta_1 rand()(P_{id} - X_{id}) + \eta_2 rand()(P_{gd} - X_{id}) \quad (2)$$

where X_{id}' represents the current positions of i particle and d dimension, X_{id} represents the previous positions of i particle and d dimension, V_{id} represents the previous velocity of i particle and d dimension, V_{id}' represents the current velocity of i particle and d dimension, P_{id} represents the individual's best position, P_{gd} represents the best position found in the whole swarm so far, $0 \le \omega < 1$ is an inertia weight, η_1 and η_2 are acceleration constants, rand() generates random number from interval [0,1]. A limit velocity calls $V_{\max}$. If calculate velocity of a particle exceeds this value, it will replace value of $V_{\max}$.

B. Mutation Operators

Mutation operators are techniques, preventing premature convergence and trapping in local optimum. They create a variation of a population. Many researchers proposed various mutation methods such as:

Higashi and Iba [9] used random number generated from Gaussian distribution to randomly apply mutation op-

erator for a particle position value of one dimension. The mutation operator uses the function:

$$mutate\ (X_{id}) = X_{id}(1 + gaussian\ (\sigma)) \quad (3)$$

Stacey et al. [10] implement a mutation operator from a Cauchy distribution.

$$mutate\ (X_{id}) = X_{id} + cauchy\ (\alpha) \quad (4)$$

C. Particle Swarm Optimization with mutation

The pseudo-code for the PSO with mutation algorithm is as follows:

1) Initialize: Generate the initial particles by randomly generating the position and velocity for each particle.

2) Evaluate: Evaluate the fitness of each particle.

3) Update Particle: Update each particle position X_{id} according to Eq. (1) and (2).

4) Update Particle best position P_{id} : For each particle, if its fitness is better than the fitness of its previous best position (P_{id}), update $P_{id} = X_{id}$.

5) Update P_{gd} : For each particle, if its fitness is better than the fitness of the best position (P_{gd}) of all particles, update P_{gd} .

6) Mutate: apply mutation operator.

7) Iterate: If the stop criterion is not satisfied or iteration less than maximum iteration, then go to Step 2, else stop.

D. When to apply mutation operator

1) Based on mutation rate [7]. Mutation is applied in each iteration after all the particles have been updated. For each particle and for each dimension of a particle conditionally mutate the particle position in this dimension equal to mutation probability.

The mutation probability is dependent on the number of particles and dimensions per particle.

$$Mutation\_\text{Pro}b = \frac{Mutaion\_rate}{Num\_particle \times Num\_Dimension} \quad (5)$$

2) Based on particles diversity [8]. Mutation is applied in each iteration after all the particles have been updated If population average distance (D(t)) is less than some threshold ($D_{\min}$). For each particle and for each dimension of particle, if a random number uniformly generated in the range [0:1] is less than P_m (P_m is random value on interval [0.1,0.3]), then mutate the particle position in this dimension.

The population average distance D(t) is described as following:

$$D(t) = \frac{1}{M \times L} \sum_{i=1}^{M} \sqrt{\sum_{d=1}^{D} (x_{id}^{t} - \overline{p}_{id})^2} \quad (6)$$

Where $\overline{P}_{id}$ is the average value of the d_{th} dimension coordinate values of all particles, D is the dimension of the solution space, M is the size of the population, L is the diagonal length of search space, X^{t}_{id} is the d_{th} dimension coordinate values of the i_{th} particle.

Normally, mutation should occur when PSO trap in local optima. Apply mutation prematurely prevent PSO from obtain optima. On the other hand, apply mutation too slow waist PSO searching time.

III. MUTATION PERIOD CALCULATION FOR PSO

Mutation should be applied when PSO trap in local optima. But, how to detect when local optima occur? It is hard to specify using some fix mutation rate. Usually local optima occur when all particles are closed to each other. But how closed? Also, diversity among particles is an average value. Even if the average is small, some particles may still be position further away.

This paper try to find average iteration period (I_p) that local optima occur. Local optima occur when fitness of best particle repeat its value for some iterations (M_r). In order to find I_p , PSO is executed for some limited iterations (1000 to 3000 iterations) and observed the number of times local optima occur (L_c).

The pseudo-code for the PSO with mutation is the same as in C except change step 6 and adds another step 8 as follow:

Step 6 Counting: if F_{cbp} is equal or worse than F_{pbp} then

N_r++

else

$N_r = 0$ and $F_{pbp} = F_{cbp}$

If N_r equal M_r then

$N_r = 0$ and L_c++

Where F_{cbp} is fitness of current best particle, F_{pbp} is fitness of previous best particle.

Step 8 mutation period Calculation: The mutation period calculation according to Eq (7).

$$I_p = \frac{\max\ generation}{L_c \times M_r} \quad (7)$$

Where N_r is number repeat, M_r is max repeat, L_c is local count. I_p is average iteration period.

Once identify the average iteration period (I_p), it can be applied in step 6. The main idea is that the local optima occurs when fitness of best particle repeat its value for some X iterations. If this X is too small it may not be the real local optima (fitness of best particle is not repeated if X is larger). So in this step, X is change from M_r to I_p in order to be certain that it is large enough and local optima really occur.

Step 6 Mutate: if F_{cbp} is equal or worse than F_{pbp} then

N_r ++

else

$N_r = 0$ and $F_{pbp} = F_{cbp}$

if N_r equal I_p then $N_r = 0$ and for each particle and for each dimension of particle: If a random number uniformly generated in the range [0:1] is less than M_p (mutation percentage), then mutate the particle dimension according to the mutation operator being used.

IV. Experiments and Result

A. Benchmark Functions

The six benchmark functions with different properties are chosen to PSO with mutation. the equations are listed below:

1) Sphere function

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Where $x \in [-5.12,5.12]^n$

Global minimum is $x^* = (0,...,0), f(x^*) = 0$

Sphere function is continues, convex and no has local minimum except the global one (single modal, uni-model).

2) Rosenbrok's function

$$f(x) = \sum_{i=1}^{n} [100(x_i^2 - x_{i+1}) + (x_i - 1)^2]$$

Where $x \in [-2.048,2.048]^n$

Global minimum is $x^* = (1,...,1), f(x^*) = 0$

Rosenbrok's function has global optimum. It is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult. Rosenbrok's function is several local minima (multimodal) or single modal.

3) Ackley's function

$$f(x) = -20 \exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$$

Where $x \in [-30,30]^n$

Global minimum is $x^* = (0,...,0), f(x^*) = 0$

Ackley's function has an exponential term that covers its surface with numerous local minima (several local minima) and regularly distributed. Ackley's function is multimodal.

4) Rastrigin's function

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

Where $x \in [-5.12,5.12]^n$

Global minimum is $x^* = (0,...,0), f(x^*) = 0$

Rastrigin's function has the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed. Rastrigin's function is multimodal.

5) Griewank's function

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

Where $x \in [-300,300]^n$

Global minimum is $x^* = (0,...,0), f(x^*) = 0$

Griewank's function is similar to Rastrigin function. It has many widespread local minima. However, the locations of the minima are regularly distributed. Griewank's function is multimodal.

6) Schwefel's function

$$f(x) = 418.9829 \times n + \sum_{i=1}^{n} (x_i \times \sin(\sqrt{|x_i|}))$$

Where $x \in [-500,500]^n$

Global minimum is $x^* = (-420.96,..,-420.96), f(x) = 0$

Schwefel's function is double sum function. Its main difficulty is that its gradient is not oriented along their axis due to the epitasis among their variables; in this way, the algorithms that use the gradient converge very slowly. Schwefel's function is multimodal.

B. Parameters Setting

Parameters are as follows for all experiments: acceleration constants of η_1 and η_2 are both set to be 1.496180 and inertia weight ω = 0.729844, as suggested by den Bergh [11], the number of population is 60. Each function has 30 dimensions. Each function runs PSO 50 times. The maximum iteration is 5000. The limited iteration is 2000. Mutation rate is 0.5. $D_{\min} = 0.001$, $M_r = 10$, $M_p = 0.02$.

C. The experiment results and comparison

Three experiments are performed to compare when to apply mutation, mutation rate [7] (MPSO), diversity [8] (DPSO) and average iteration period (IPSO).

The pseudo-code for mutation operators of all 3 methods is as follows:

If a random number generated uniformly in the range [0:1] is less than 0.5, then mutate the particle dimension by using the Eq. (8). Else mutate the particle dimension by using the Eq. (9).

$$mutate(X_{id}) = X_{id} - X_{id} \times cauchy(\alpha) \quad (8)$$

$$mutate(X_{id}) = X_{id} + X_{id} \times cauchy(\alpha) \quad (9)$$

Where X_{id} represents the current positions of i particle and d dimension, Cauchy (α) denote Cauchy random numbers with the scale parameter of 1.

TABLE I. RESULTS FROM THE EXPERIMENT FIND AVERAGE ITERATION PERIOD. AVERAGE OF L_c OVER 50 RUNS, I_p AND ESTIMATION I_p.

Function	L_c	I_p	estimation I_p
Sphere	0.035	5714.286	5714
Rosenbrok	1.03	19.40805	19
Ackley	88.5	2.261037	2
Rastrigin	115.71	1.728459	2
Griewank	75.55	2.647253	3
Schwefel	131.86	1.51676	2

TABLE II. THE MAXIMUN (MAX), MININUM (MIN) AND AVERAGE (AVG) BESTFITNESS OVER 50 RUNS. RESULT FROM THE EXPERIMENTS ARE PERFORMED TO COMPARE WHEN TO APPLY MUTATION.

Function		PSO	MPSO	DPSO	IPSO
Sphere	Min	4.43E-122	8.86E-119	2.62E-15	1.32E-122
	Avg	6.75E-115	1.81E-113	3.76E-14	5.35E-115
	Max	2.79E-113	2.88E-112	2.43E-13	2.54E-113
Rosenbrok	Min	0	5.92E-29	0.000118	0
	Avg	1.31E-29	7.19E-27	0.00057	7.00E-30
	Max	9.86E-29	1.62E-25	0.0012	8.38E-29
Ackley	Min	7.11E-15	7.11E-15	2.31E-08	7.11E-15
	Avg	0.712812	9.81E-15	1.01E-07	7.67E-15
	Max	3.93204	2.13E-14	2.88E-07	1.42E-14
Rastrigin	Min	44.7731	0	0	0
	Avg	101.267	3.18385	21.2531	1.08012
	Max	166.157	15.9192	83.5762	37.9098
Griewank	Min	0	0	0	0
	Avg	0.011604	0.012402	0.00713	0.005611
	Max	0.065853	0.051408	0.0612	0.044153
Schwefel	Min	572.455	473.754	710.63	0.007506
	Avg	1653.81	1215.18	1189.12	228.391
	Max	2783.33	1895.01	1895.01	710.683

From the experimental results of Table I show that multi-model functions have a little mutation period. Rosenbrok's function has larger mutation period. Sphere's function has the largest mutation period.

All the function has minimal point at zero. Hence any methods that yield solution nearer to the zero point are the better method. From the experimental results of Table II, MPSO can increase performance for all multi-model function but decrease performance for Sphere's function and Rosenbrok's function when compare to PSO. DPSO can increase performance for some multi-model function but decrease performance for Rosenbrok's function and Sphere's function when compare to PSO. IPSO performed better than FPSO and PPSO for all multi-model function and Sphere's function. IPSO performed better than PSO for all multi-model function.

V. CONCLUSION

The mutation operator is applied to avoid local trapping in PSO. This paper proposed when to apply this mutation. If apply too early, PSO may not reach the optimal yet. On the contrary, apply too slow wait searching time. From the experimental results, it is shown that the proposed method yield better optimal point for all test functions.

VI. REFERENCES

[1] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, IEEE International Conference on Neural Networks, pp.1942-1948, 1995.

[2] R. C. Eberhart and J. Kennedy, A New Optimizer Using Particle Swarm Theory, Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp.39-43, 1995.

[3] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on Evolutionary Computation, vol. 8,no. 3, pp. 240–255, 2004.

[4] R. A. Krohling, Gaussian particle swarm with jumps, in Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK,pp. 1226-1231, 2005.

[5] R. A. Krohling, L. dos Santos Coelho, PSO-E: Particle Swarm with Exponential Distribution , in Proceedings of the IEEE Congress on Evolutionary Computation, pp1428- 1433, July 2006.

[6] Muhammad Imran, Hajira Jabeen, Mubashir Ahmad, Qamar Abbas,Waqas Bangyal, Qamar Abbas. Opposition based PSO and Mutation Operators , 201O 2nd International Conference on Education Technology and Computer (ICETC), pp506- 508.

[7] Andrews P S. An investigation into mutation operators for particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Vancouver, Canade: IEEE, 2006. 1044¡1051

[8] MO Lin, ZHENG Hua. Improved PSO Algorithm with Adaptive Inertia Weight and Mutation, 2009 World Congress on Computer Science and Information Engineering, pp.622-625, CSIE.2009.428

[9] N. Higashi and H. Iba. Particle swarm optimization with Gaussian mutation, Proc.of the 2003 IEEE Swarm Intelligence Symphosium, pp. 72-79, 2003.

[10] A. Stacey, M. Jancic, and I. Grundy.Particle swarm optimization with mutation, Proc. of the 2003 IEEE Congr. On Evol. Comput., pp. 1425-1430,2003.

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