A Hybrid Particle Swarm Algorithm with Cauchy Mutation

Hui Wang
School of Computer Science
China University of Geosciences
Wuhan, 430074 China
wanghui_cug@yahoo.com.cn

Yong Liu
University of Aizu
Tsuruga, Ikki-machi, Aizu-Wakamatsu
Fukushima 965-8580 Japan
yliu@u-aizu.ac.jp

Changhe Li
School of Computer Science
China University of Geosciences
Wuhan, 430074 China
lchwfx@yahoo.com.cn

Sanyou Zeng
School of Computer Science
China University of Geosciences
Wuhan, 430074 China
sanyou-zeng@263.net

***Abstract*— Particle Swarm Optimization (PSO) has shown its fast search speed in many complicated optimization and search problems. However, PSO could often easily fall into local optima because the particles could quickly get closer to the best particle. At such situations, the best particle could hardly be improved. This paper proposes a new hybrid PSO (HPSO) to solve this problem by adding a Cauchy mutation on the best particle so that the mutated best particle could lead all the rest of particles to the better positions. Experimental results on many well-known benchmark optimization problems have shown that HPSO could successfully deal with those difficult multimodal functions while maintaining fast search speed on those simple unimodal functions in the function optimization.**

Ⅰ. INTRODUCTION

Particle Swarm Optimization (PSO) was firstly introduced by Kennedy and Eberhart in 1995 [1]. It is a simple evolutionary algorithm which differs from other evolutionary algorithms in which it is motivated form the simulation of social behavior. PSO has shown good performance in finding good solutions to optimization problems [2], and turned out to be another powerful tool besides other evolutionary algorithms such as genetic algorithms [3].

Like other evolutionary algorithms, PSO is also a population-based search algorithm and starts with an initial population of randomly generated solutions called particles [4]. Each particle in PSO has a position and a velocity. PSO remembers both the best position found by all particles and the best positions found by each particle in the search process. For a search problem in an n-dimensional space, a potential solution is represented by a particle that adjusts its position and velocity according to Eqs. (1) and (2):

$$V_i^{(t+1)} = w * V_i^{(t)} + c_1 * rand1() * (P_i - X_i^{(t)}) + c_2 * rand2() * (P_g - X_i^{(t)}) \quad (1)$$

$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)} \quad (2)$$

where X_i and V_i are the position and velocity of particle i, P_i and P_g are previous best particle for the ith particle and the global best particle found by all particles so far respectively, and w is an inertia factor proposed by Shi and Eberhart [5], and rand1() and rand2() are two random numbers independently generated within the range of [0,1], and c_1 and c_2 are two learning factors which control the influence of the social and cognitive components.

One problem found in the standard PSO is that it could easily fall into local optima in many optimization problems. Some research has been done to tackle this problem [6-8]. One reason for PSO to converge to local optima is that particles in PSO can quickly converge to the best position once the best position has no change in a local optimum. When all particles become similar, there is little hope to find a better position to replace the best position found so far. In this paper, a new hybrid PSO (HPSO) is proposed. HPSO uses an idea from fast evolutionary programming (FEP)[9] to mutate the best position by Cauchy mutation. It is to hope that the long jump from Cauchy mutation could get the best position out of the local optima where it has fallen. HPSO has been tested on both unimodal and multi-modal function optimization problems. Comparison has been conducted between HPSO and another improved PSO called FDR-PSO [10]. HPSO has also been compared to other evolutionary algorithms, such as classical EP (CEP) and FEP [9].

The rest of the paper is organized as follows: Section 2 describes the new HPSO algorithm. Section 3 lists benchmark functions used in the experiments, and gives the experimental settings. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes with a summary and a few remarks.

Ⅱ. HPSO ALGORITHM

Some theoretical results have shown that the particle in PSO will oscillate between their pervious best particle and the global best particle found by all particles so far before it converges [11-12]. If the searching neighbors of the global best particle would be added in each generation, it would extend the search space of the best particle. It is helpful for the whole particles to move to the better positions. This can be accomplished by having a Cauchy mutation on the global best particle in every generation. The one-dimensional Cauchy density function centered at the origin is defined by:

$$f(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < \infty \tag{3}$$

where t>0 is a scale parameter [13]. The Cauchy distributed function is

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{t}\right) \tag{4}$$

The reason for using such a mutation operator is to increase the probability of escaping from a local optimum [9]. The Cauchy mutation operator used in HPSO is described as follows:

$$W(i) = \left(\sum_{j=1}^{popsize} V[j][i]\right) / \ popsize \tag{5}$$

where V[j][i] is the *i*th velocity vector of the *j*th particle in the population, popsize is the population size. W(i) is a weight vector within [$-W_{max}$, W_{max}], and W_{max} is set to 1 in this paper.

$$P_g'(i) = P_g(i) + W(i) * N(X_{min}, X_{max}) \tag{6}$$

where N is a Cauchy distributed function with the scale parameter t=1, and N(X_{min} , X_{max}) is a random number within [X_{min} , X_{max}], which is a defined domain of a test function. The main steps of the HPSO algorithm are as follows:

HPSO algorithm

while *iteration* < = *max-iterations* **do**
begin
for each particle i **do**
begin
  Calculate fitness value
  if the fitness value is better than its best fitness value in history
  then Update P_i
  if the fitness value is better than the best fitness value in history
  then Update P_g
  Calculate particle velocity according equation (1)
  Update particle position according to equation (2)
end
for j = 1 to the population size **do**
begin
  Update W[j] according to equation (5)
  if fabs(W[j] > W_{max}) **then** W[j] = W_{max}
end
Mutate P_g according to equation (6) for *N* times
Select a best particle P_g^{best} from the *N* particles after having *N* mutations
if the fitness value of P_g^{best} is better than P_g
then $P_g = P_g^{best}$
end

III. Benchmark Problems and Experimental Settings

10 well-known test functions used in [9], [10] have been chosen in our experimental studies. The purpose is not to show that HPSO is better than any other improved PSO algorithms, but to explain that the idea of FEP is very useful for improving the performance of the standard PSO.

The 10 test functions used in our experiments are listed in Tables 1 and 2. They are high-dimensional problems, in which functions f_1 to f_6 in Table 1 are unimodal functions, and functions f_7 to f_{10} in Table 2 are multimodal functions. All the functions used in this paper are to be minimized.

Table 1. The 6 unimodal functions used in our experimental studies, where n is the dimension of the functions, f_{min} is the minimum values of the function, and $X \subseteq R^n$ is the search space.

Test Function	n	X	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	20	$[-5.12, 5.12]$	0
$f_2(x) = \sum_{i=1}^{n} i * x_i^2$	20	$[-5.12, 5.12]$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	20	$[-65.536, 65.636]$	0
$f_4(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	20	$[-1, 1]$	0
$f_5(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)^2]$	30	$[-30, 30]$	0
$f_6 = \sum_{i=1}^{n} i * x_i^4 + random[0,1)$	30	$[-1.28, 1.28]$	0

Table 2. The 4 multimodal functions used in our experimental studies, where n is the dimension of the functions, f_{min} is the minimum values of the function, and $X \subseteq R^n$ is the search space.

Test Function	n	X	f_{min}
$f_7 = \sum_{i=1}^{n} -x_i * \sin(-\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	−12569.5
$f_8 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$f_9 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
$f_{10} = -20 * \exp\left(-0.2 * \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]$	0

The selection of the parameters w, c_1, c_2 of Eq. (1) is very important. It can greatly influence the performance of PSO algorithms and its variations, By following the suggestions given in [14], c_1, c_2 and w are set in Table 3. However, there are a few differences for different problems. In this paper, V_{max} in all related PSO algorithms is set to 2.0, and W_{max} and N are set to 1 and 20 respectively.

Table 3. The specific parameter settings

Test Function	Number of Generations	Popsize	$c_1 = c_2$	w	W_{max}	N
f_1- f_4	1000	10	1.49618	0.72984	1	20
f_5	10000	50	1.49618	0.72984	1	20
f_6	3000	50	1.49618	0.72984	1	20
f_7	5000	50	1.49618	0.72984	1	20
f_8	5000	50	1.49618	0.72984	1	20
f_9	1000	50	1.49618	0.72984	1	20
f_{10}	1000	50	1.49618	0.72984	1	20

In order to compare the different algorithms, the same settings have been used in FDR-PSO, CEP and FEP in our experiments. In HPSO and FDR-PSO, the same maximal generations and the same population size were used. In HPSO, CEP and FEP, less or the same maximal generations were used.

Ⅳ. EXPERIMENTAL RESULTS

Two comparisons have been conducted in this section. One is among HPSO, the standard PSO, and FDR-PSO [10]. The other is among HPSO, PSO, CEP and FEP [9].

A. Comparisons among HPSO, PSO, and FDR-PSO

Table 4 shows the comparisons among HPSO, PSO, and the best FDR-PSO in [10] for functions f_1 to f_4. All results are averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation and "Minima" shows the minimum in the algorithm. It is obvious that HPSO performs better than both the standard PSO and FDR-PSO. From the results on f_3, it could be seen that HPSO could find much better solutions. It suggests that the Cauchy mutation used in HPSO could speed up the search process.

B. Comparisons among HPSO, PSO, CEP, and FEP

The average results of HPSO, PSO, CEP, and FEP on functions f_5 to f_{10} over 50 runs are given in Table 5. On unimodal functions f_5 and f_6, HPSO has shown the fastest convergence among 4 tested algorithms. On multimodal functions f_7 to f_{10}, HPSO performed much better than both the standard PSO and CEP. It suggests HPSO is less likely to fall into local optima compared to the standard PSO and CEP. HPSO could perform equally well as FEP on f_7 and f_9, better than FEP on f_{10}, but worse than FEP on f_8.

The significant improvement achieved by HPSO can be contributed to the search ability of Cauchy mutation operator, which extends the search space of the best particle. Such extended neighbor search space will greatly help particles move to better positions. In some cases, the extended neighbors have included the global optima. Therefore, HPSO had reached better solutions than the standard PSO.

Table 4. The results achieved for f_1 to f_4 using different algorithms

Function	HPSO		PSO		FDR-PSO[10]
	Mean Best	Std Dev	Mean Best	Std Dev	Minima
f_1	1.79e-7	3.51e-7	1.57e-6	5.11e-6	2.63e-7
f_2	6.38e-7	1.98e-6	3.53e-6	1.55e-5	1.07e-5
f_3	0.398	0.3082	17.5646	36.4659	0.9080
f_4	2.53e-19	9.38e-19	1.84e-19	6.14e-19	5.3e-19

Table 5. The results achieved for f_5 to f_{10} using different algorithms

Function	HPSO		PSO		CEP [9]		FEP [9]	
	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
f_5	1.419	1.4256	1.8016	2.8389	6.17	13.61	5.06	5.87
f_6	4.37e-3	1.51e-3	4.57e-3	1.69e-3	1.8e-2	6.4e-3	7.6e-3	2.6e-3
f_7	-12558.9	6.2373	-6736.5	544.5	-7917.5	634.5	-12554.5	52.6
f_8	31.8005	9.1618	37.0721	9.7295	89.0	23.1	4.6e-2	2.1e-3
f_9	3.66e-2	3.19e-2	8.96e-2	0.2882	8.6e-2	0.12	1.6e-2	2.2e-2
f_{10}	8.86e-6	8.58e-2	1.1289	1.1298	9.2	2.8	1.8e-2	2.1e-3

In order to find more differences between HPSO and the standard PSO, Figure 1 shows the evolution process of the mean of function values of the populations for HPSO and PSO. For the simple unimodal functions, HPSO and PSO performed equally well at the beginning because the particles at that time are not good enough so that both methods could improve well. Once the particles in the populations are close to the best particle, the convergence of PSO becomes slower because the search steps in PSO become smaller. It can be seen in Eq. (1) that the search steps are generally larger when the particles are further away from the best particle, while they become smaller when the particles get closer to the best particle. With the help of Cauchy mutation on the best particles, HPSO could move the best particle away from the rest of particles in the population so that the fast speed could remain through the whole evolution process. For the difficult multimodal functions f_5 and f_7, Cauchy mutation on the best particle could move the best particle away from the local minimum once the best particle falls into it. Because of such mutations made on the best particle, HPSO could successfully find better solutions while maintaining fast search speed. On the other hand, PSO could be easily tracked into local minima without the mutation done on the best particle.

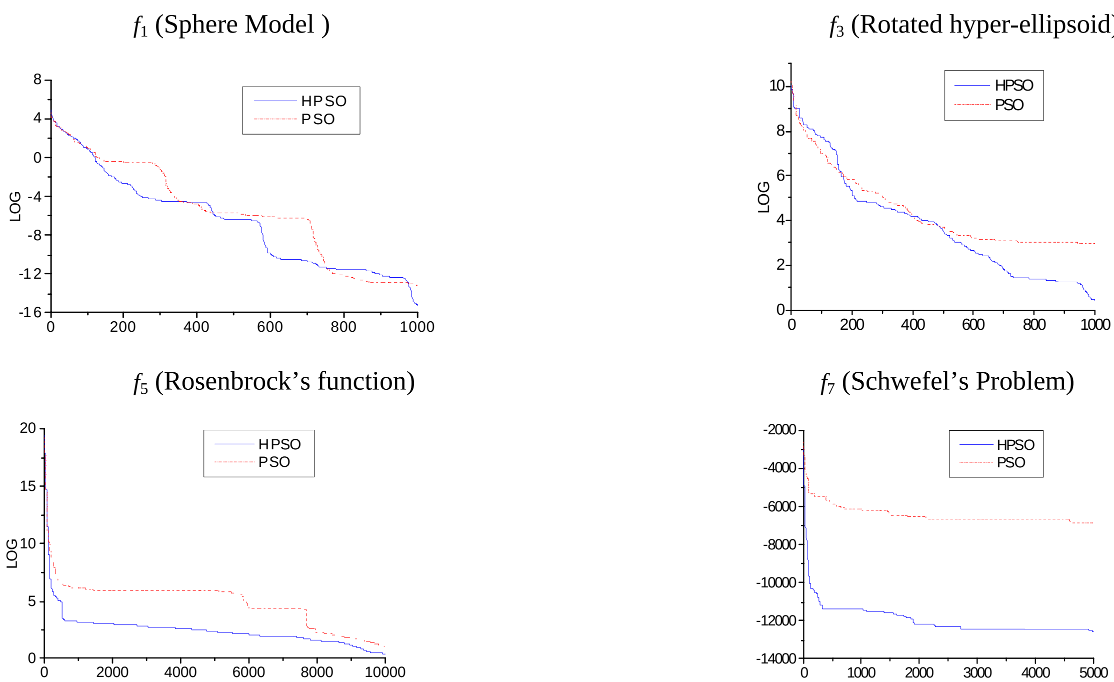

Fig. 1. Comparison between PSO and HPSO on f_1, f_3 , f_5 and f_7 . The horizontal axis is the number of generations and the vertical axis is the function value.

C. Number of Replacements

To investigate how effective the Cauchy mutation operator used in HPSO is, the average number of replacements over 50 runs is given in Table 6. Each replacement happens when the mutated P_g' is better than P_g. The results have shown that the replacements had happened oftener. On the multimodal functions and some difficult unimodal functions such as f_3, while the replacements had seldom taken place on simple unimodal functions. The reason is that PSO has more chances to fall into local minima for those difficult multimodal functions so that it would need more Cauchy mutations in order to move the best particles away from the local minima.

Table 6. The results are averaged over 50 runs, where "Number of Replacements" indicates the average number of replacements in the Cauchy mutation operator.

Function	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Number of Replacements	19.87	17.27	63.1	21.1	84.17	62.6	372.9	47.73	109.43	70.63

V. Conclusions

The idea of HPSO is to use a Cauchy mutation operator derived from FEP [9] to help PSO avoid local optima. By applying a Cauchy mutation on the best particle found by all particles so far in each generation, HPSO could find better solutions than PSO.

HPSO has been compared with the standard PSO, an improved FDR-PSO, CEP, and FEP on both 6 unimodal functions and 4 multimodal functions. The results have shown that HPSO could have faster convergence on those simple unimodal functions, and better global search ability on those multimodal functions compared to the standard PSO. However, there are still fewer cases where HPSO had fallen in the local optima as what had happened on HPSO for the function f_8. It suggests that a Cauchy mutation on the best particle alone might not be enough to prevent the search from falling in the local optima. As the Cauchy mutation introduced on the best particle, such mutation could be applied on all particles in the populations. It is expected that the Cauchy mutation on all particles could perform better for those extremely hard multimodal function optimization problems.

References

[1] J. Kennedy and R. Eberhart, Particle Swarm Optimization, IEEE International Conference on Neural Networks, Perth, Australia. 1995.

[2] K. E Parsopoulos, V. P. Plagianakos, G. D. Magoulas, M. N. Vrahatis, Objective Function "stretching" to Alleviate Convergence to Local Minima, Nonlinear Analysis TMA 47, 3419-3424, 2001.

[3] R. Eberhart and Y. Shi, Comparison between Genetic Algorithms and Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA. 1998

[4] X.. Hu, Y. Shi and R. Eberhart, Recentt Advenes in Particle Swarm, Congress on Evolutionary Computation, Portland, Oregon, June 19-23, 90-97, 2004

[5] Y. Shi and R. Eberhart, A Modified Partilce Swarm Optimzer, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ. 69-73, 1998.

[6] F. van den Bergh, A. P. Engelbrecht, Cooperative Learning in Neural Networks using Particle Swarm Optimization, South African Computer Journal, 84-90, Nov. 2000.

[7] X, Xie, W, Zhang, Z, Yang, Hybrid Particle Swarm Optimizer with Mass Extinction, International Conf. on Communication, Circuits and Systems (ICCCAS), Chengdu, China. 1170-1174, 2002.

[8] M. Lovbjerg, T. Krink, Extending Particle Swarm Optimisers with Self-Organized Criticality, Proceedings of Fourth Congress on Evolutionary Computation, vol. 2, 1588-1593, 2002.

[9] X. Yao, Y. Liu and G. Lin, Evolutionary Programing Made Faster, IEEE Transacations on Evolutionary Computation, vol. 3, 82-102, July 1999.

[10] K. Veeramachaneni, T. Peram, C. Mohan, L. A. Osadciw, Optimization Using Particle Swarms with Near Neighbor Interactions, Proc. Genetic and Evolutionary Computation (GECCO 2003), vol. 2723,110-121, 2003.

[11] E. Ozcan and C. K. Mohan, Particle Swarm Optimization: Surfing the Waves, Proceedings of Congress on Evolutionary Computation (CEC1999), Washington D.C., 1939-1944, 1999.

[12] F. van den Bergh, A. P. Engelbrecht, Effect of Swarm Size on Cooperative Particle Swarm Optimizers, Genetic and Evolutionary Computation Conference, San Francisco, USA, 892-899, 2001.

[13] W. Feller, An Introduction to Probability Theory and Its Applications, volume 2, John Wiley & Sons, Inc., 2nd edition, 1971.

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