An Improved Probability Particle Swarm Optimization Algorithm

Qiang Lu[1] and Xuena Qiu[2]

[1] School of Automation, Hangzhou Dianzi University,
Hangzhou, 310018, China
lvqiang@hdu.edu.cn
[2] School of Telecommunication, Ningbo University of Technology,
Ningbo, 310018, China
qiuxn26@hotmail.com

Abstract. This paper deals with the problem of unconstrained optimization. An improved probability particle swarm optimization algorithm is proposed. Firstly, two normal distributions are used to describe the distributions of particle positions, respectively. One is the normal distribution with the global best position as mean value and the difference between the current fitness and the global best fitness as standard deviation while another is the distribution with the previous best position as mean value and the difference between the current fitness and the previous best fitness as standard deviation. Secondly, a disturbance on the mean values is introduced into the proposed algorithm. Thirdly, the final position of particles is determined by employing a linear weighted method to cope with the sampled information from the two normal distributions. Finally, benchmark functions are used to illustrate the effectiveness of the proposed algorithm.

Keywords: Normal distribution, probability particle swarm optimization, evolutionary computation.

1 Introduction

The particle swarm optimization (PSO) algorithm, originally proposed by Kennedy and Eberhart (1995) [1,2], has become one of the fascinating branches of evolutionary computation. The underlying motivation for the development of the PSO algorithm is the social behavior of animals such as bird flocking and fish schooling. In the last decade, the PSO algorithm has been well studied [3,4,5]. For example, in [3], a quantum-behaved particle swarm optimization algorithm was proposed for a class of unconstrained functions by introducing several simple quantum concepts. In [4,5], the authors designed a class of hybrid optimization approaches by combining the PSO algorithm with other methods such as an ant colony algorithm in order to improve the efficiency of the PSO algorithm. These improved PSO algorithms [3,4,5] have a common feature, i.e., the particle's velocity part is considered. However, in 2003, Kennedy [6] (2003) proposed a bare

Y. Tan, Y. Shi, and K.C. Tan (Eds.): ICSI 2010, Part I, LNCS 6145, pp. 102–109, 2010.

bones particle swarm algorithm where the particle's velocity part is dropped and the position of any particle on each dimension satisfies a normal distribution with mean $\frac{p_{id}+p_{gd}}{2}$ and standard deviation $|p_{id} - p_{gd}|$ where p_{gd} and p_{id} are the global best position and the previous best position, respectively. From the bare bones particle swarm algorithm, one can see that this algorithm provides another way, i.e., probability method to improve the search performance of the PSO algorithm. Recently, the study on probability particle swarm algorithm has received increasing interest from scholars [7,8,9,10,11]. For instance, in [7,8], a particle swarm optimization algorithm based on pheromone mechanism was proposed for unconstrained functions by introducing an information-shared matrix, which is used to store useful information. In [9,10], based on Gaussian or Cauchy method, the bare bones particle swarm algorithm was further improved. Moreover, in order to correct the weakness of the PSO algorithm such as linearization of the curve attained in steady-state, Secrest [11] (2003) presented a Gaussian particle swarm optimization algorithm. However, only a few results on the study of the probability particle swarm algorithm are reported. Therefore, how to further improve the search performance of the probability particle swarm algorithm for the problem of unconstrained optimization is the motivation of the current study.

In this paper, in order to deal with the problem of unconstrained optimization, we will propose an improved probability particle swarm optimization (IPPSO) algorithm. The proposed algorithm uses two normal distributions to update the position of each particle. The means of two normal distributions are the global best position and the previous best positions, respectively. Moreover, we will introduce a disturbance to the means of two normal distributions. And the information about fitness will be used as the standard deviation of the two normal distributions. We will use twelve benchmark unconstrained functions to illustrate the effectiveness of the proposed algorithm.

2 An Improved Probability Particle Swarm Optimization Algorithm

2.1 Algorithm Details

By sampling two normal distributions $N((randn()\eta + 1)p_{gd}(k), u_1(k+1))$ and $N((randn()\eta+1)p_{id}(k), u_2(k+1))$, two temporal positions $\xi^g_{id}(k+1)$ and $\xi^i_{id}(k+1)$ are obtained by using (1) and (2), respectively.

$$\xi^g_{id}(k+1) = \frac{randn()}{u_1(k+1)} + (randn()\eta + 1)p_{gd}(k) \tag{1}$$

$$\xi^i_{id}(k+1) = \frac{randn()}{u_2(k+1)} + (randn()\eta + 1)p_{id}(k) \tag{2}$$

where $randn()$ is a random number satisfying a normal distribution with mean 0 and variance 1; η is a heuristic parameter; p_{id} denotes the previous best position

of the i^{th} particle; p_{gd} is the global best position among particles; $u_1(k+1)$ and $u_2(k+1)$ are adjustment parameters, which can be calculated by

$$u_1(k+1) = u_1(k) + f(x_i(k)) - f(p_g(k)) \tag{3}$$
$$u_2(k+1) = u_2(k) + f(x_i(k)) - f(p_j(k)) \tag{4}$$

where $x_i(k)$ denotes the position of the i^{th} particle; $f : R^n \rightarrow R$ denotes the fitness function; $u_1(k)$ and $u_2(k)$ can be evaporated as

$$u_1(k) = evp \cdot u_1(k) \tag{5}$$
$$u_2(k) = evp \cdot u_2(k) \tag{6}$$

A threshold l_{th} is defined. If one random number in [0,1] is less than the threshold l_{th}, the position of the particle will be updated based on (8). Otherwise, it will be adjusted according to (7).

$$x_{id}(k+1) = c_1\xi^g_{id}(k+1) + c_2\xi^i_{id}(k+1) \tag{7}$$
$$x_{id}(k+1) = rand() \tag{8}$$

where $rand()$ denotes a random value in the search range; c_1 and c_2 are adjustment parameters and satisfy $c_1 + c_2 = 1$; $x_{id}(k+1)$ is a new position of the i^{th} particle.

2.2 Algorithm Procedure

The procedure of the IPPSO algorithm is summarized as

Algorithm 1 ***(IPPSO Algorithm)***

1. Initialization.
   (a) Initialize the threshold l_{th}, adjustment parameters c_1 and c_2, and heuristic parameter η.
   (b) Initialize the adjustment parameters u_1 and u_2 with 0.
2. Repeat until a given maximal number of iteration is achieved.
   (a) Evaluate the fitness of each particle.
   (b) Store the global best particle and fitness.
   (c) Store the previous best particle of the i^{th} $(i = 1, 2, \cdots n)$ particle and its fitness.
   (d) Perform (5) and (6).
   (e) Each particle performs the following steps.
      i. Perform (3) and (4).
      ii. Perform (1) and (2).
      iii. If a random number is lesser than the threshold l_{th}, perform (8) and (7) otherwise.

Table 1. Parameters of the IPPSO algorithm

Population	l_{th}	c_1	c_2	$(\eta, evp) d < 10$	$(\eta, evp) 10 \leq d < 600$	Max iteration
20	0.01	0.8	0.2	(0,0.01)	(0.5,0.9)	2000

3 Simulated Experiments

In order to evaluate the performance of the IPPSO algorithm, the experiments will be conducted from two aspects: the fixed number of function evaluations and the fixed convergence precision of functions. Several algorithms published in recent years such as BBPSO [6], BBPSO-GJ1 [9], BBPSO-CJ2 [10] and GPSO [11] are used as comparison examples. The parameters of the IPPSO algorithm are listed in Table 1. Moreover, twelve benchmark unconstrained functions can be seen in [5].

3.1 Fixed Number of Function Evaluations

The objective of this subsection is to evaluate the performance of the IPPSO algorithm based on the fixed number of function evaluations, which is set as 40000. The average objective function values of the BBPSO, GPSO, BBPSO-GJ1 and BBPSO-GJ2 algorithms over 50 runs on each benchmark function ($f_1 - f_{12}$) are shown in Table 2 and 3.

From Table 2 and 3, one can see that the better average values on functions f_1, f_2, f_3, f_4, f_8 and f_9 are obtained by using the IPPSO algorithm in contrast to other algorithms. Moreover, for functions f_5 and f_6, the BBPSO algorithm can obtain better function values while the BBPSO-GJ2 algorithm can gain the better function values for functions f_7, f_{10}, f_{11} and f_{12}.

3.2 Fixed Convergence Precision of Functions

In this subsection, success rate, the number of function evaluations and standard deviation are used as indexes to evaluate the performance of the IPPSO algorithm under the fixed convergence precision of functions. The results of these algorithms on functions $f_1 - f_{12}$ (each algorithm runs 50 times on each function) are reported in Table 4 and 5.

From Table 4 and 5, one can see that on functions $f_1 - f_8$ and f_{12}, 100% success rate is gained by the IPPSO algorithm while on functions $f_9 - f_{10}$ 100% success rate is only obtained by the BBPSO-GJ2 algorithm. For functions f_8 and f_{11}, function values obtained by the BBPSO, BBPSO-GJ1 and GPSO algorithms cannot reach the given convergence precision within the max iteration. For the number of function evaluations, compared with other algorithms, the IPPSO algorithm also gains the less number of function evaluations on all functions except for f_3, f_{10}, f_{11} and f_{12}.

Table 2. Results of the IPPSO, GPSO and BBPSO algorithms based on the fixed number of function evaluations

Function	Average objective function value (Std.)		
	IPPSO	*GPSO*	*BBPSO*
f_1	3.0000(0)	4.08(5.3446)	3.0000(2.57E-15)
f_2	0.3979(0)	0.3979(4.59E-16)	0.3979(3.36E-16)
f_3	-2.0000(0)	-1.9128(6.94E-2)	-1.9976(1.71E-2)
f_4	-186.7309(0)	-186.7309(1.02E-13)	-186.7309(1.785E-13)
f_5	-3.8627(3.94E-5)	-3.8628(2.57E-15)	-3.8628(4.93E-15)
f_6	-3.2798(5.60E-2)	-3.2651(6.02E-2)	-3.2842(5.62E-2)
f_7	8.44E-11(3.37E-11)	2.4632E+4(6.87E+3)	3.69E-24(7.31E-24)
f_8	2.22E-12(5.67E-13)	3.4661(2.1744)	174.805(147.1863)
f_9	8.824(0.99)	40.0857(92.2618)	100.557(318.9685)
f_{10}	3.4E-3(6.8E-3)	9.9E-3(9.8E-3)	1.22E-2(1.64E-2)
f_{11}	1.67(6.22)	124.1306(27.52)	73.6069(23.6241)
f_{12}	5.41E-7(4.33E-7)	2.71E-2(2.14E-2)	1.4E-2(2.06E-2)

Table 3. Results of the IPPSO, BBPSO-GJ1 and BBPSO-GJ2 algorithms based on the fixed number of function evaluations

Function	Average objective function value (Std.)		
	IPPSO	$BBPSO-GJ2$	$BBPSO-GJ1$
f_1	3.0000(0)	3.0000(2.68E-15)	3.0136(5.12E-2)
f_2	0.3979(0)	1.2059(1.0143)	0.3979(6.08E-5)
f_3	-2.0000(0)	-2.0000(0)	-1.9882(2.23E-2)
f_4	-186.7309(0)	-186.5531(2.08E-1)	-186.6962(1.538E-1)
f_5	-3.8627(3.94E-5)	-3.8595(3.2E-3)	-3.8413(2.92E-2)
f_6	-3.2798(5.60E-2)	-3.0103(7.18E-2)	-2.9864(1.55E-1)
f_7	8.44E-11(3.37E-11)	0(0)	6.52E+4(5.49E+4)
f_8	2.22E-12(5.67E-13)	2.2614E-6(1.01E-5)	1.25E+4(4.78E+3)
f_9	8.824(0.99)	9.5279(0.6023)	1.44E+5(4.14E+5)
f_{10}	3.4E-3(6.8E-3)	0(6.34E-3)	2.63E-2(3E-2)
f_{11}	1.67(6.22)	1.1689(4.006)	27.343(8.288)
f_{12}	5.41E-7(4.33E-7)	0(0)	5.2E-3(1.6E-2)

Table 4. Results of the IPPSO, GPSO and BBPSO algorithms based on the fixed convergence precision of functions

Function	Required accuracy	Success rate			Function evaluations(Std.)		
		IPPSO	*GPSO*	*BBPSO*	*IPPSO*	*GPSO*	*BBPSO*
f_1	1E-3	100	100	100	329 (136.91)	304 (87.27)	329 (71.85)
f_2	1E-3	100	100	100	152 (76.45)	287 (133.75)	275 (126.07)
f_3	1E-3	100	46	94	1028 (632.75)	21842 (1.98E+4)	3026 (9.45E+3)
f_4	1E-2	100	100	98	373 (148.87)	668 (262.14)	2368 (6.45E+3)
f_5	1E-4	100	100	100	538 (342.27)	1736 (1352)	356.8 (86.125)
f_6	1E-3	100	44	68	402 (79.61)	22709 (1.97E+4)	1317 (1.85E+4)
f_7	1E-5	100	0	100	1810 (101.67)	40000 (0)	14172 (897.80)
f_8	1E-5	100	0	0	9077 (3.11E+3)	40000 (0)	40000 (0)
f_9	10	98	0	62	1614 (5.54E+3)	40000 (0)	19488 (1.77E+4))
f_{10}	1E-2	84	56	58	11043 (1.56E+4)	26188 (1.25E+4)	22424 (1.51E+4)
f_{11}	1E-2	90	0	0	14362 (1.25E+4)	40000 (0)	40000 (0)
f_{12}	1E-5	100	30	74	5156 (6.32E+3)	30137 (1.57E+4)	13479 (1.7E+4)

Table 5. Results of the IPPSO, BBPSO-GJ1 and BBPSO-GJ2 algorithms based on the fixed convergence precision of functions

Function	Required accuracy	Success rate			Function evaluations		
		IPPSO	BBPSO -GJ1	BBPSO -GJ2	IPPSO	BBPSO -GJ1	BBPSO -GJ2
f_1	1E-3	100	82	100	329 (136.91)	8845 (1.57E+4)	698 (349.44)
f_2	1E-3	100	100	96	152 (76.45)	457 (708.70)	4795 (9.12E+3)
f_3	1E-3	100	36	100	1028 (632.75)	25980 (1.89E+4)	628 (182.23)
f_4	1E-2	100	82	18	373 (148.87)	8936 (1.55E+4)	37066 (7.86E+3)
f_5	1E-4	100	10	0	538 (342.27)	36027 (1.2E+4)	40000 (0)
f_6	1E-3	100	0	0	402 (79.61)	40000 (0)	40000 (0)
f_7	1E-5	100	0	100	1810 (101.67)	40000 (0)	11498 (604.88)
f_8	1E-5	100	0	86	9077 (3.11E+3)	40000 (0)	32108 (5.98E+3)
f_9	10	98	0	100	1614 (5.54E+3)	40000 (0)	4523 (528.48)
f_{10}	1E-2	84	0	100	11043 (1.56E+4)	40000 (0)	9103 (804.56)
f_{11}	1E-2	90	0	100	14362 (1.25E+4)	40000 (0)	9041 (925.17)
f_{12}	1E-5	100	10	100	5156 (6.32E+3)	36495 (1.07E+4)	1554 (412.36)

4 Conclusion

An improved probability particle swarm optimization (IPPSO) algorithm has been proposed. We have also illustrated the effectiveness of the proposed algorithm through twelve benchmark functions from two aspects: the fixed number of function evaluations and the fixed convergence precision of functions. It is worth mentioning that BBPSO-GJ2 can also generate much better results, especially for functions with many local maxima.

Acknowledgments. This work was supported in part by the Natural Science Foundation of Zhejiang under Grant Y1090426, and by the Technical Project of Zhejiang under Grant 2009C33045.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Network, pp. 1942–1948. IEEE Press, New York (1995)
2. Eberhart, R.C., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: 6th International Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Press, New York (1995)
3. Lu, Q., Chen, R.-Q., Yu, J.-S.: Quantum Continuous Particle Swarm Optimization Algorithm and Its Application. System Engineering-Theory and Practice 28, 122–130 (2008)
4. Liu, B., Wang, L., Jin, Y.-H.: Improved Particle Swarm Optimization Combined with Chaos. Chaos, Solitons and Fractals 25, 1261–1271 (2005)
5. Shelokar, P.S., Siarry, P., Jayaraman, V.K.: Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continuous Optimization. Applied Mathematics and Computation 188, 129–142 (2005)
6. Kennedy, J.: Bare Bones Particle Swarms. In: Swarm Intelligence Symposium, pp. 80–87. IEEE Press, New York (2003)
7. Lu, Q., Qiu, X., Liu, S.: A Discrete Particle Swarm Optimization Algorithm with Fully Communicated Information. In: ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 393–400. ACM Press, New York (2009)
8. Lu, Q., Liu, S., Qiu, X.: Design and Realization of Particle Swarm Optimization Based on Pheromone Mechanism. Acta Automatica Sinica 35, 1410–1419 (2009)
9. Krohiling, R.A.: Gaussian Particle Swarm with Jumps. In: IEEE Congress on Evolutionary Computation, pp. 1226–1231. IEEE Press, New York (2005)
10. Krohiling, R.A., Mendel, E.: Bare Bones Particle Swarm Optimization with Gaussian or Cauchy jumps. In: IEEE Congress on Evolutionary Computation, pp. 3285–3291. IEEE Press, New York (2009)
11. Secrest, B.R., Lamont, G.B.: Visualizing Particle Swarm Optimization-Gaussian Particle Swarm Optimization. In: Swarm Intelligence Symposium, pp. 198–204. IEEE Press, New York (2003)