Novel Particle Swarm Optimization for Unconstrained Problems

Peifeng Wu[1], Jianhua Zhang[1]

1. School of Electrical and Electronic Engineering, North China Electric Power University, Beijing 102206
E-mail: wuzheng820807@yahoo.cn

Abstract: Estimation of Distribution Algorithm (EDA) is a class of evolutionary algorithms which construct the probabilistic model of the search space and generate new solutions according to the probabilistic model. Particle Swarm Optimization (PSO) is an algorithm that simulates the behavior of birds flocks and has good local search ability. This paper proposes a combination (EDAPSO) of EDA with PSO for the global optimization problems. The EDAPSO proposed in this paper combines the exploration of EDA with the exploitation of PSO. EDAPSO can perform a global search over the entire search space with faster convergence speed. EDAPSO has two main steps. First, the algorithm generates new solutions according to the probabilistic model. Then, EDAPSO updates the whole population according to improved velocity updating equation. EDAPSO has been evaluated on a series of benchmark functions. The results of experiments show that EDAPSO can produce a significant improvement in terms of convergence speed, solution accuracy and reliability.

Key Words: Exploration, Exploitation, Convergence speed

1 INTRODUCTION

Particle swarm optimization (PSO) has been proposed by Kennedy and Eberhart in [1]. PSO simulates the population behavior of birds and is easy to implement. The algorithm has better robustness than many algorithms such as GA, is less sensitive to the nature of the problems and can be used for function optimization [2], control system [3], parameters optimization [4] and neural network[5]. However, similar to other population-based evolutionary algorithms, basic PSO may get trapped in local optima when solving complex multimodal problems Therefore, avoiding local optima and accelerating convergence speed of algorithm have become the two most important goals in the research of PSO. Many versions of improved PSO have been proposed to achieve the above two goals. In [6], an intelligent move mechanism was introduced to solve high-dimensional problems. Shi and Eberhart introduced a linearly decreasing weight to balance global and local search abilities[7]. Liuetal. proposed the center particle swarm optimization (Center PSO) to improve the performance of LDWPSO [8].Higashi and Iba introduced Gaussian mutation operator to increase the diversity of the population[9].

In this paper, we propose a novel version of PSO (EDAPSO). To avoid premature convergence of PSO and to accelerate the convergence speed, we combine the exploration of EDA with the exploitation of PSO. In EDAPSO, new solutions are sampled from a probability model and then the whole population updates according to the improved PSO with a novel velocity updating equation. This paper provides a comprehensive set of experimental verifications of our proposed algorithm.

2 PARTICLE SWARM OPTIMIZATION ALGORITHM

PSO is a population based optimization algorithm. The algorithm is initialized with a population of random particles in search space and searches for optima by updating direction vectors and velocity vectors. In n-dimensional search space, the direction vector of the i-th particle can be represented by vector $X_i = (x_{i1}, x_{i2}, \cdots x_{in})$ and the velocity vector of the i-th particle can be represented by another vector $V_i = (v_{i1}, v_{i2}, \cdots v_{in})$. At each generation, the particles update their velocities and directions according to the following two equations:

$$v_{id}^{(t+1)} = \omega v_{id}^{(t)} + c_1 r_1 (p_{id}^{(t)} - x_{id}^{(t)}) + c_2 r_2 (p_{gd}^{(t)} - x_{id}^{(t)}) \quad (1)$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \quad (2)$$

where ω is called inertia weight. This factor has the function of balancing exploration and exploitation. If the factor is increased, the exploration of the algorithm will be enhanced. The exploitation will be enhanced with the decreasing of the factor. r_1 and r_2 are independently uniformly distributed random number with range (0,1). c_1 and c_2 are positive constant parameters called learning factors. $p_{id}^{(t)}$ is the best solution obtained by the i-th particle and $p_{gd}^{(t)}$ is the best solution obtained by the whole population.

3 EDAPSO

There are two main steps in most population-based optimization algorithms. They are generating population and updating population. Basic PSO generates uniformly

978-1-4673-5534-6/13/$31.00 ©2013 IEEE

distributed random population and updates population according to velocity updating equation and direction updating equation.

However, PSO doesn't have mechanism to obtain the global information about the whole search space. This case may make the algorithm get trapped in local optima. After researching EDA[10], [11], we incorporate the idea of EDA into PSO to generate solutions which are more promising than those solutions generated by the PSO, and consequently, to explore the search space more effectively.

EDAPSO updates the population with an improved velocity updating equation which can accelerate convergence speed. The procedure of EDAPSO is given as follows:

Step1:Generate M uniformly distributed random solutions to form initial population pop(0)

Step2:Select e promising solutions from the current population to form elite set E(k) and build probabilistic model P(k) according to Eqs. (3)-(4)

$$p(x) = \prod_{i=1}^{e} N(x_i; \mu_i^k, \sigma_i^k) \tag{3}$$

$$N(x_i; \mu_i^k, \sigma_i^k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{1}{2}(\frac{x_i-\mu_i}{\sigma_i})^2} \tag{4}$$

In Eq.(3), we use n univariate and independent normal distributions as the joint probability distribution. The mean $\hat{\mu}_i^k$ and the standard deviation $\hat{\sigma}_i^k$ of the i-th variable at k-th generation are the two parameters need to be estimated.

The two parameters can be estimated as follows:

$$\hat{\mu}_i^k = \frac{1}{e} \sum_{k=1}^{e} x_{ik} \tag{5}$$

$$\hat{\sigma}_i^k = \frac{1}{e} \sum_{k=1}^{e} (x_{ik} - \hat{\mu}_i)^2 \tag{6}$$

Where e is the size of E(k) .

Step3:Save the better half of the solutions in pop(k) and the other half of the solutions are sampled according to the constructed probabilistic model P(k).

Step4:Use PSO with improved updating equation to update pop(k). The velocity updating equation is as follow:

$$v_{id}^{(t+1)} = \omega v_{id}^{(t)} + 2\omega_1 \times (p_{gd} - x_{id}^{(t)}) \tag{7}$$

Step5:If the given stopping criterion is not met, k=k+1, goto step 2.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Benchmark Functions for Comparison

In order to verify the performance of EDAPSO, we compare the performance of EDAPSO with basic PSO, LDWPSO and CFPSO. The four algorithms are applied to settle a comprehensive set of benchmark functions which has 11 different optimization problems[2]. The functions are presented as follows:

The first is Rosenbrock function, defined as:

$$f_1 = \sum_{i=1}^{N-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{8}$$

With $-100 \le x_i \le 100$, the best solution of f_1 is $x^* = (1, 1, ..., 1)$ and $f(x^*) = 0$.

$$f_2 = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{9}$$

With $-100 \le x_i \le 100$, the best solution of $-0.5 \le x_i \le 0.5$ is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_3 = \sum_{i=1}^{N} (\sum_{j=1}^{N} x_j)^2 \tag{10}$$

With $-100 \le x_i \le 100$, the best solution of f_3 is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_4 = \sum_{i=1}^{N} x_i^2 + (\sum_{i=1}^{N} 0.5 i x_i)^2 + (\sum_{i=1}^{N} 0.5 i x_i)^4 \tag{11}$$

With $-100 \le x_i \le 100$, the best solution of f_4 is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_5 = \sum_{i=1}^{N} |x_i| + \prod_{i=1}^{N} |x_i| \tag{12}$$

With $-100 \le x_i \le 100$, the best solution of f_5 is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_6 = \sum_{i=1}^{N} (|x_i + 0.5|)^2 \tag{13}$$

With $-100 \le x_i \le 100$, the best solution is $-0.5 \le x_i \le 0.5$ and best function value is 0.

$$f_7 = \sum_{i=1}^{N} |x_i \sin(x_i) + 0.1 x_i| \tag{14}$$

With $-100 \le x_i \le 100$, the best solution of f_7 is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_8 = \sum_{i=1}^{N} x_i^2 \tag{15}$$

With $-5.12 \le x_i \le 5.12$, the best solution of f_8 is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_9 = \sum_{i=1}^{N} i x_i^2 \tag{16}$$

With $-5.12 \le x_i \le 5.12$, the best solution of f_9 is $x^* = (0, 0, ..., 0)$

$$f_{10} = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{17}$$

With $-100 \le x_i \le 100$, the best solution of f_{10} is $x^* = (0, 0, ..., 0)$ and $f(x^*) = 0$

$$f_{11} = \sum_{k=1}^{N} [\sum_{i=1}^{N} (i^k + 0.5)(\frac{x_i}{i})^k - 1]^2 \quad (18)$$

With $-N \le x_i \le N$, the best solution of f_{11} is $x^* = (1, 2, ..., N)$ and $f(x^*) = 0$

4.2 Comparison Strategies and Metrics:

We measure the number of function calls (NFCs) which is commonly used metric in literature [12-13] to compare the convergence speed. If the NFCs of an algorithms is smaller than other algorithms, this algorithm has higher convergence speed .The termination criterion is to find a value smaller than VTR(value to reach)before reaching the maximum number of function calls (MAX_{NFC}).In order to compare the convergence speed with NFC more directly, we introduce the parameter acceleration rate(AR). AR is defined as follows:

$$AR_{\frac{\text{algorithm A}}{\text{algorithm B}}} = \frac{NFC_{\text{algorithm A}}}{NFC_{\text{algorithm B}}} \quad (19)$$

Where AR > 1 means the convergence speed of algorithm b is faster than the speed of algorithm a.

In the experiments, we also use success rate(SR) to compare the performance of the algorithms. SR for each test function is defined as follows:

$$SR = \frac{number\ of\ times\ reached\ VTR}{total\ number\ of\ trials} \quad (20)$$

To present the convergence speed of different problems and the robustness of the algorithms, we introduce the average acceleration rate AR_{ave} and the average success rate SR_{ave} over n test functions. The two parameters are calculated as follows

$$AR_{ave} = \frac{1}{n} \sum_{i=1}^{n} AR_i \quad (21)$$

$$SR_{ave} = \frac{1}{n} \sum_{i=1}^{n} SR_i \quad (22)$$

The parameter settings in the experiments are showed as follows:

population size: M=100

Basic PSO: $c_1 = c_2 = 2$ [1]

LDWPSO: $\omega_{\min} = 0.4$, $\omega_{\max} = 0.9$ [7]

CFPSO: $\omega = 0.729$, $c_1 = c_2 = 1.49445$ [14]

Maximum NFCs (MAX_{NFC}) = 10^6

Value to reach (VTR)= 10^{-6}

To make the comparisons reliable and fair, for all the experiments: (1)the presented values are the average of for 30 independent runs.(2) in EDAPSO , the extra number of function calls which is required for the promising particles sampled according to the constructed probabilistic model are counted.

4.3 Comparison EDAPSO with basic PSO, LDWPSO and CFPSO

We compare EDAPSO with three classical PSO (basic PSO, LDWPSO and CFPSO). Table 1 shows the results of solving 11 functions. The results of the experiments show NFCs, SR for each function . The last row of the table shows the average success rates and the average acceleration rate over 11 functions. The functions in this section are low dimensional functions. The parameter e is set to be 0.3M(M is the size of particles in population). Table.1 is too long .Thus, we divide it into two parts. The average success rates of PSO, LDWPSO, CFPSO and EDAPSO are 0.6483, 0.7, 0.7363 and 0.8696, respectively. The four algorithms fail to solve f_{10} (Rastrigin's function)..Basic PSO,LDWPSO and CFPSO all fail to solve f_2 .The comparison of average success rates reveals that EDAPSO has the best robustness among the four algorithms.

$AR_{\frac{PSO}{LDWPSO}}$ is 0.97, which means the convergence speed of PSO is 3% faster than the convergence speed of LDWPSO. $AR_{\frac{PSO}{CFPSO}}$ is 1.04,which means the convergence speed of CFPSO is 4% faster than the convergence speed of PSO. $AR_{\frac{PSO}{EDAPSO}}$ is 9.25, which means the convergence speed of EDAPSO is 825% faster than the convergence speed of PSO.

$AR_{ave\frac{LDWPSO}{EDAPSO}} = \frac{AR_{ave\frac{PSO}{EDAPSO}}}{AR_{ave\frac{PSO}{LDWPSO}}} = 9.53$, which means the convergence speed of EDAPSO is 853% faster than the convergence speed of LDWPSO.

$AR_{ave\frac{LDWPSO}{EDAPSO}} = \frac{AR_{ave\frac{PSO}{EDAPSO}}}{AR_{ave\frac{PSO}{LDWPSO}}} = 8.89$, which means convergence speed of EDAPSO is 789% faster than the convergence speed of CFPSO. The comparisons of average acceleration rate indicate that EDAPSO has the fastest

Table1. Comparison of Four Algorithms for Low-dimensional Problems

function	dimension	NFC_P	NFC_L	NFC_C	NFC_E	SR_P	SR_L
f_1	10	798047	812587	532583	148058	0.333	0.2
f_2	10	-	-	-	11757	0	0

f_3	10	210732	152137	183659	26553	0.566	1
f_4	10	153351	148239	132274	40350	1	1
f_5	10	138734	167823	156327	19202	1	1
f_6	10	117628	127089	104976	8250	1	1
f_7	10	233745	311937	406239	17257	0.466	0.5
f_8	10	118839	108752	108836	7357	1	1
f_9	10	131625	120137	116534	8422	1	1
f_{10}	10	-	-	-	-	0	0
f_{11}	10	165729	234894	196728	70382	0.766	1
AVE						0.6483	0.7

function	dimension	SR_C	SR_E	$AR\frac{PSO}{LDWPSO}$	$AR\frac{PSO}{CFPSO}$	$AR\frac{PSO}{EDAPSO}$
f_1	10	0.733	0.933	0.98	1.49	5.39
f_2	10	0	0.633	-	-	-
f_3	10	1	1	1.38	1.14	7.93
f_4	10	1	1	1.03	1.15	3.80
f_5	10	1	1	0.82	0.88	7.22
f_6	10	1	1	0.92	1.12	11.25
f_7	10	0.466	1	0.74	0.57	13.54
f_8	10	1	1	1.09	1.09	16.15
f_9	10	1	1	1.09	1.12	15.62
f_{10}	10	0	0	-	-	-
f_{11}	10	0.9	1	0.7	0.84	2.35
AVE		0.7363	0.8696	0.97	1.04	9.25

convergence speed among the four algorithms. The comparison shows that EDAPSO has the best robustness and the fastest convergence among the four algorithms. The optimization ability is the best among the four algorithms on the whole.

5 CONCLUSION

In this paper, the search idea of EDA has been employed to collect global information. An improved velocity updating equation was used to accelerate the speed of the algorithm. Embedding these two improvements, we proposed EDAPSO.

PSO, LDWPSO, CFPSO and EDAPSO were compared in terms of robustness and convergence speed. The comparisons for SR (success rates) and AR (accelerate rates) show that EDAPSO has the fastest convergence speed and best robustness among the four algorithms.

Utilizing the idea of EDA to obtain more global information and improving the solution accuracy of the problems is a new method. Further studies are still required to investigate its advantages, disadvantages and limitations. This paper can be viewed as a first step in this direction. Possible directions for future work include adaptive setting of e (size of elite set) or applying the idea of EDA to other popular algorithms.

REFERENCES

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Network, 1942-1948, 1995.

[2] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transaction on Evolutionary Computation. Vol.10, No.3, 281-295, 2006.

[3] H.M, Feng, Particle swarm optimization learning fuzzy systems design. Proceedings-3rd International Conference on Information Technology and Applications,363-365, 2005.

[4] A. A. A. Esmin, A. R. Aoki, G. L. Torres, Particle swarm optimization for fuzzy membership functions optimization, Proceedings of the International Conference on Syetem, Man and Cybernetics,106-111,2002.

[5] Z. He, C. Wei, L.Yang,etal, Extracting rules from fuzzy neural network by particle swarm optimization, Proceedings of the IEEE Conference on Evolutionary Computation,74-77, 1998

[6] S. Y. Ho, H. S. Lin, W. H. Liauh, and S. J. Ho, OPSO: Orthogonal particle swarm optimization and its application to task assignment problems, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, Vol.38, No.2,288-298, 2008

[7] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation, 69-73,1998.

[8] Y. Liu, Z. Qin, Z. Shi and J. Lu, Center particle swarm optimization, Neurocomputing, Vol.70, No.4-6, 672-679, 2007.

[9] N. Higashi and H. Iba, Particle swarm optimization with gaussian mutation, Proceedings of the IEEE Swarm Intelligence Symposium, 72-79, 2003.

[10] M.Enokizono,Y.Akinari, Estimation of current distribution by a hybrid of genetic algorithm and sampled pattern matching method, IEEE Transactions on Magnetics, Vol.31, No.3, 2012-2015, 1995.

[11] M. Pelikan, D.E. Goldberg, F. Lobo, A survey of optimization by building and using probabilistic models, Computational Optimization and Applications. Vol .1,No.3,111-128, 2002.

[12] J. Andre, P. Siarry, and T. Dognon, An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, Advance in Engineering Software, Vol.32, No.1, 49-60, 2001.

[13] S. Rahnamayan and M.M.A.Salama, Opposition based differential evolution algorithms, 2006 IEEE Congress on Evolutionary Computation, 2010-2017, 2007:

[14] M.Clrec, J. Kennedy, The particle swarm-explosion stability and Convergence in a multidimensional complex space. IEEE Transaction on evolutionary computation,Vol.6, No.1: 58-73, 2002.