Available online at www.sciencedirect.com

SciVerse ScienceDirect

Physics Procedia 24 (2012) 1369 – 1376

Physics Procedia

2012 International Conference on Applied Physics and Industrial Engineering

Incremental Particle Swarm Optimization

Xiaohua Xu[1], Zhoujin Pan[2], Yanqiu Xi[2], Ling Chen[2]

Department of Computer Science
Yangzhou University
Yangzhou 225009, China

Abstract

By simulating the population size of the human evolution, a PSO algorithm with increment of particle size (IPPSO) was proposed. Without changing the PSO operations, IPPSO can obtain better solutions with less time cost by modifying the structure of traditional PSO. Experimental results show that IPPSO using logistic model is more efficient and requires less computation time than using linear function in solving more complex program problems.

Key words:PSO; particle size; logistic model; computational cost

1. Introduction

The particle swarm optimization (PSO) is developed by Kennedy J. and Eberhart R. C. [1] [2] based on bird flocking behavior in 1995[3]. It has been recognized as one of the computational intelligence techniques intimately related to evolutionary algorithms and now attracted the interest of researchers all around the world. Recently PSO has gained widespread achievement in solving variety optimization problems [4][5][6][7]. It has been widely used in function optimization, neural network training, pattern classification, fuzzy system control and as well as other engineering field [12]. However, PSO has no strict mathematical foundation as a novel optimization search algorithm. A lot of problems remain to be further studied.

As the PSO algorithm convergences very fast in the early stage of particle swarm optimization and in the latter period of the evolution, its convergence speed [10] becomes slower and the precision of the solution could not be improved rapidly, the academia made a lot of improvement of the algorithms with extensive attention. To improve the performance of the algorithm, most the researchers pay attention to the inertia weight and the constraint factor but less efforts are paid to reform the structure of the algorithm. In this paper, a new type of PSO named IPPSO (Increment particles-size PSO) with increment size of the

1875-3892 © 2011 Published by Elsevier B.V. Selection and/or peer-review under responsibility of ICAPIE Organization Committee.
Open access under CC BY-NC-ND license. doi:10.1016/j.phpro.2012.02.204

particles is presented by modifying the structure of the classical PSO. Our IPPSO uses PSO as a subroutine [13].

In our IPPSO algorithm, we use the experience from the previous swarm after several iterations to expand the size of particle swarm. In this case, the new particles should adopt the existing experience as much as possible. At first, we initialize the particles with random positions and velocities. Then we execute the original PSO algorithm for several iterations to gain the experience to distinguish the poorer particles with better ones. Using this experience we enlarge the size of the particle swarm, update the parameters, and loop to the criterion. The results show clearly that IPPSO with logistic model can have higher probability to obtain optimal solution with less cost than traditional PSO on more complex functions, while the IPPSO with linear function can have a better solution on those functions that have only one optimal solution.

2. Framework of the algorithm

The algorithm of Original PSO: Let s, ts, te be the particle swarm size, the number of minimum iterations and the number of the maximum iterations respectively.

ALGORITHM PSO(s, ts , te)
1. t ← ts
2. Initialize particles of size s with random positions and velocities in the search space.
#please note that while in the IPPSO we should put this step outside.
3. ***While*** (t ≤ te or criterion did not meet) ***do***
4. t ← t + 1
5. Evaluate the desired optimization fitness function of s particles.
6. Compare s particles' fitness evaluation with pbesti:
pbesti← current value
pi← current location xi
If current value is better than pbesti
7. Identify the best successful particle with so far, and assign its index to the variable pg.
8. Update the velocity and position of the s particles according to the following equation:
vi ← w*vi + U(0,φ1)*(pi − xi)+ U(0,φ2)*(pg − xi)
xi ← xi + vi
9. ***End while***
Notes:
-w represents the inertia weight [9]
-U (0, φi) represents a vector of random numbers uniformly distributed in [0, φi] which is randomly generated at each iteration and for each particle.
-Each component of *vi* is kept within the range [−Vmax, +Vmax][8]

PSO algorithm convergences very fast in the early stage of particle swarm optimization and in the latter period of the evolution, its convergence speed becomes slower and the precision of the solution could not be improved rapidly. In addition, thought out the procedure of the particles' flight to the optimum location, the size of particles remains un-changing. Considering of that, in this paper we propose a new PSO algorithm IPPSO in which the size of particles will in-crease with a function S(i). Simply when newcomers join a team, the old members will share their experience to them. After learning the good and giving up the bad, newcomers will be able to contribute to the team as soon as they can, so that the team would become more efficient.

To gain the experience appropriately, we initialize the new particles' location near the area of the particles whose fitness are better than the remaining with a variation parameter representing the bad experience.

The function s(t). s(t) is the function of particle size which is changing in different iterations. The selection of s(t) influences the quality of the solutions in IPPSO. When we choose the function of s(t), the procedure of human population evolution can be used for reference. Since among all species, human being is the prominent one. The evolution of humankind is just an excellent procedure of optimization. In our experiment, we choose two kinds of functions to be s(t): linear equations and logistic model.

Figure 1 shows an simple example of a linear function: s(t)=t.

In 1837 Verhulst, a bio-mathematician in Holland, suggest the logistic model of population size. The model is based on one basic assumption that population growth rate is on a linear decreasing function of population.

Let r(x) is the function of population growth rate, xmax be max population that the environment can sustain, and r is the original population growth rate. The modeling procedure is listed follow:

As the assumptions we have:

$$\begin{cases} r(x) = r - sx \\ r(0) = r \\ r(x\max) = 0 \end{cases} \tag{1}$$

Then:

$$r(x) = r(1 - \frac{x}{x\max}) \tag{2}$$

So put r(x) into the exponential growth of differential equations:

$$\begin{cases} \frac{dx}{dt} = rx(1 - \frac{x}{x\max}) \\ x(0) = x_0 \end{cases} \tag{3}$$

Its solution is:

$$x(t) = \frac{x\max}{1 + (\frac{x\max}{x_0} - 1)e^{-rt}} \tag{4}$$

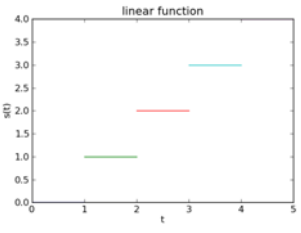

Figure 1. Linear incremental function

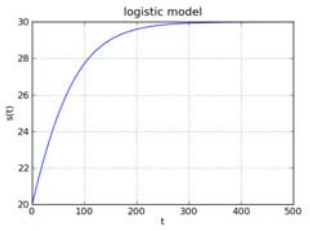

Figure 2. Logistic model

Figure 2 shows an simple example of logistic model,in which x_0=20,r=-0.018, max=30. Experimental results show that IPPSO can get higher efficiency when logistic model is used to choose proper s.

IPPSO is a framework based on the PSO which are denoted as PSO and be used as a subroutine after a little modification (pay attention to step (2) in IPPSO. We use the experience from the previous swarm after several iterations to expand the size of particle swarm. In this case, the new particles should adopt the existing experience as much as possible. Using PSO as a subroutine, the algorithm IPPSO is described as follows.

ALGORITHM IPPSO(S,T):
1. i← 0
2. Initialize particles of size si with random positions and velocities in the search space.
2. PSO (Si, Ts, Te)
3. **While** (criterion did not meet) **do**
4 i ← i + 1
5. Update PSO's parameters //such as the S(i) and the locations of new particles
6. PSO (Si, Ts, Te)
7. **End while**

3. The experimental results

We test our algorithms using Core Duo T2250 PC with 1.73 GHZ and 1.25G main memory under ubuntu 9.10 with python 2.6 platform. To illustrate that the performance of our algorithm is higher than traditional PSO, we tested them on 4 benchmarks of non-linear programming problems. Each of the problems is presented by its functional form, domain and optimum with its three-dimensional image or three-dimensional projection image and local amplification image near the optimal value. We tested 100 times for each function and recorded the average of the optimal solutions and the average execution time.

3.1 Function #1

$$f(x) = \sum_{i=1}^{2} x_i \sin(\sqrt{|x_i|}), x_i \in [-512, 512]$$

$$\min f(-420.9687, 420.9687) = -837.965774544868$$

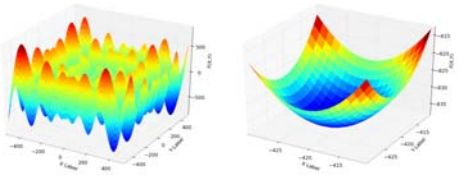

Figure 3. funtion #1 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	-755.058940209	-776.278102968	-737.28866316	-778.746574771
average time(s)	0.318322372437	0.469437122345	0.19641354558	0.331097235124

3.2 Funtion #2

$$f(x)=\sum_{i=1}^{n}x_i^2, x_i\in[-5.120,5.120]$$

$$\min\ f(0,.....,0)=0$$

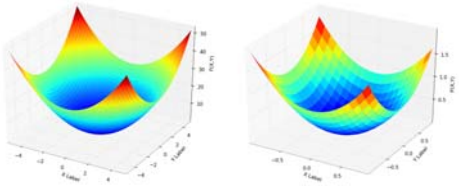

Figure 4. funtion #2 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	2.51626336901e-07	3.97565689249e-07	4.78945181961e-08	2.13070540958e-06
average time(s)	0.0627581477165	0.0977435112	0.118504285812	0.0425280094147

3.3 Funtion #3

$$f(x)=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1.0+0.001(x_1^2+x_2^2)]^2}, x_i\in[-100,100]$$

$$\min\ f(0,0)=0$$

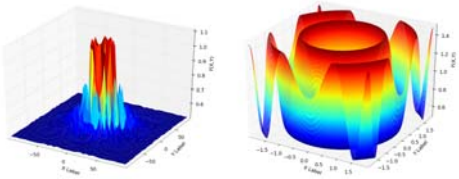

Figure 5. funtion #3 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	0.0671466761873	0.120217388016	0.0634140830675	0.0597353504211
average time(s)	8.4149137338	25.0273016294	0.574889802933	0.259199619293

3.4 Function #4

$$f(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|, x_i\in[-10,10]\ \min\ f(0,...,0)=0$$

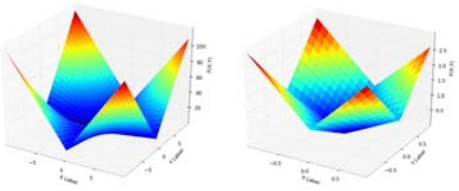

Figure 6. funtion #4 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	3.0876439496e-07	6.76193538968e-07	2.74031457751e-08	4.566014478e-06
average time(s)	0.0415429115	0.469437122	0.04428959369	0.04297959804

3.5 Function #5

$$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7, x_i\in[-50,50] \quad \min f(0,0)=0$$

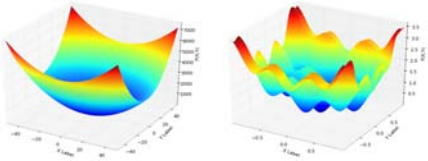

Figure 7. function#5 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	4.1293206878e-06	7.23162239313e-08	1.62277759008e-08	1.499333689e-07
average time(s)	0.282690382004	0.436868882179	0.332519006729	0.20372760295

3.6 Function #6

$$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4, x_i\in[-5,5]$$

$$\min f(0.08983,-0.7126)=-1.031685$$

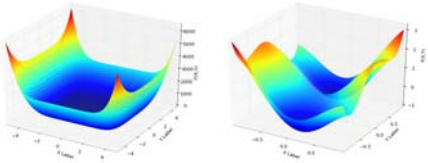

Figure 8. function#6 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average	-0.95001023665	-1.0316283246	-0.9500116599	-0.9499853089

solutions				
average time(s)	0.059343910217	0.092040634155	0.07237749099	0.043390488624

3.7 Function #7

$$f(x) = -\sin x_1 \cdot \sin^{20}(\frac{1}{\pi}x_1^2) - \sin x_2 \cdot \sin^{20}(\frac{2}{\pi}x_2^2), x_i \in [0,\pi]$$

$$\min f(2.20290552, 1.57079632) = -1.8013034100\ 99$$

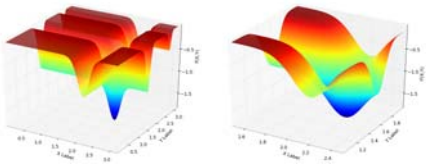

Figure 9. function#7 and local amplification image near optimal value

Algorithm name	PSO		IPPSO	
Size	50	100	Linear function	LogisticModel
Average solutions	-1.26091210804	-1.2609123263	-1.8013034034	-1.80127810327
average time(s)	0.160266470909	0.245122790337	0.29551281929	0.184376502037

Results of experiments (see Figure 10) on function #1, #3, #7 and functions #7 show that IPPSO with logistic model can do well on complex functions and the IPPSO with linear function can have a better solution on those functions that have only one optimal solution such as function #2, #4 and function #5.

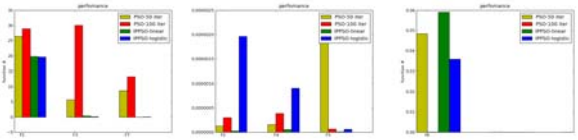

Figure 10. The result which we named it performance is calculated by the function: $\varepsilon * |p_s - p_t| * t$. ε is zoom parameter,p_s is the average solution,pt is the theoretial value while t is average execute time.

4. Conclusion

By modifying the framework of the original PSO, a new type of PSO named IPPSO with increment size of particle swarm is presented. Using a modified PSO as its subroutine, IPPSO adjusts its particle size in each stage of the evolution process. Our experimental results show that the solution by IPPSO is almost better than PSO. IPPSO can get a better solution than PSO with accelerated convergence speed .In addition, IPPSO using logistic model can gain a better performance than using linear function in solving some more complex functions. But till now we have not find a mathematical method of looking for a function of enlarge the size of the swarm. In addition, except experience we havn't find instructive means to set each incremental step with iterations follow. Our next task is getting a theoretical method guiding us finding out the function maybe the functions to perfect our method.

Acknowledgment

This research was supported in part by the Chinese National Natural Science Foundation under grant No. 60773103, and Natural Science Foundation of Education Department of Jiangsu Province under contract 09KJB20013.

References

[1]R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. An overview. Swarm Intelligence, 1(1):33～57, 2007.

[2]R. Poli. Analysis of the publications on the applications of particle swarm optimisation. Journal of Artificial Evolution and Applications, Article ID 685175, 10 pages, 2008.

[3]Shi Y and Eberhart R. Parameter Selection in Particle Swarm Optimization. In Evolutionary Programming VII: Proceedings of EP'98, pages 591–600, 1998.

[4]Shi Y, Eberhart R C. Empirical study of particle swarm optimization A . Proceeding of Congress on Evolutionary Computation C . : Piscatawa y, NJIEEE Service Center , 1999. 1945～1949.

[5]Clerc M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization A . Proceedings of the Congress on Evolutionary Computation C . Piscatawa y, NJ : IEEE Service Center , 1999. 1951～1957.

[6]Angeline P. Using selection to improve particle swarm optimization A . Proceedings of IJCNN99 C .Washington , USA , 1999. 84～89.

[7]Parsopoulos K E, Vrahatis M N. Particle swarm optimizer in noisy and continuously changing environments A . Hamza M H. Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing C . Cancun ,Mexico: IASTED/ACTA Press, 2001. 289～294.

[8]Van den Bergh F. An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, 2002.

[9]Shi Y and Eberhart R. A Modified Particle Swarm Optimizer. In Proceedings of theIEEE International Conference on Evolutionary Computation, pages 69～73, 1998.

[10]van den Bergh F, Engelbrecht A P. Using cooperative particle swarm optimization to train product unit neural networks R. IEEE International Joint Conference on Neural Networks, Washington D C, USA , 2001.

[11]D. N. Wilke, S. Kok, and A. A. Groenwold, Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity, International Journal for Numerical Methods in Engineering, Vol. 70, No. 8, pp. 962～984, 2007.

[12]Mehmet Çunkaş; M. Yasin Özsağlam. A Comparative Study On Particle Swarm Optimization and Genetic Algorithms For Traveling Salesman Problems.Cybernetics and Systems: An International Journal, 1087-6553, Volume 40, Issue 6, 2009, Pages 490 ～507

[13]XU Xiao-hua,CHEN Ling,CHEN Hong-jian.Genetic Algorithm with Variable Population Size. Journal of System Simulation, Vol.18, No.4, 2006, pages 870～876