Algorithm and Implementation of particle swarm optimization

Robin Dai, Chao Liang and Wen Zhang

Department of Mathematical Sciences, Michigan Technological University

Abstract:

We have studied and implemented the particle swarm optimization (PSO) algorithm and other improved versions. Based on our analysis and understanding of PSO, we developed a novel algorithm which tried to overcome the problem of the unstable velocity. The proposed PSO is based on constricted coefficients PSO by letting the parameters changed by logistic function over the time. In our study, we found out that the selection of parameters is problem-dependent, which is crucial to the behavior of the algorithm. We had different results by setting up different parameter. Secondly, we tried to develop a novel PSO algorithm, However, Results are hard to be compared due to the stochastic result. Each time the program gave us the significantly different result for the large dimension problem, which also indicates that high dimension optimization problem is hard to solve when the population size can not be large enough in practices. Because when search space is huge, PSO loses its power given a limit population size. In our testing, constricted coefficients PSO works better for SIAM function Inertia weighted PSO works better for Rosenbrock function.

Background:

1. Introduction

First we give some related concepts and philosophies about PSO which stands for particle swarm optimization. It is a computational optimization method which improves the candidate solutions iteratively with regard to a given measure. PSO optimizes a problem by having a bunch of candidate solutions, which are called particles, and moving these particles around in the search-space according to given formulae with respect to the particle's position and velocity. Each particle's movement is influenced by its local best position and it is led to the best known positions in the search area, which are replaced by some better positions that are found by other particles. It is expected to move the swarm to the best solutions.

PSO is originally attributed to Kennedy and Eberhart [1] and was first pointed out to simulate social behavior[3] as a stylized representation of the movement of organisms in a bird flock or fish school. This algorithm was simplified and was observed to achieve optimization. PSO is a metaheuristic as it makes few or no assumptions about the optimization problem and can search a good amount of candidate solutions，whereas PSO does not guarantee to find an optimal solution. What is more particular is that PSO does not use the gradient of the problem, which means that PSO does not require the optimization problem to be differentiable as is required by standard methods such as gradient descent or quasi-newton methods. Therefore PSO can also be used on optimization problems that are partially irregular. The above are some characters and advantages of PSO over other methods.

In our implementation, we used the original PSO, inertia weighted PSO and constricted coefficient PSO respectively to solve the test problems which include 2D SIAM function, 2D Rosonbrock function and high Dimensional Rosonbrock function. During the process of using these three methods, we found that each method has its own advantages and disadvantages. Further analysis and comparisons about these three methods will be given in the following of this report. Finally, we managed to develop another kind of PSO scheme, which is called logitPSO to solve these functions and got satisfied results. We will also give some details about this improved PSO method later.

The outline of this report is as follows: we will first describe the original PSO, inertia weighted PSO and constricted coefficient PSO briefly. After comparing the limitation of each method, we will then give the derived logitPSO algorithm. Further illustrative examples will be given, followed which we give the conclusions and discussions finally.

2. Original PSO

PSO consists of a swarm of particles each of which resides at a position in the search space. The fitness of each particle represents the quality of its position. The particles fly over the search space with a certain velocity. Both direction and speed of each particle are influenced by its own best position found so far. They are also influenced by the best solution that was found so far by its neighbors. However, PSO does not guarantee an optimal solution is ever found. In order to describe the original PSO, we use the following notations:

x := position; $\mathbf{v}$:= velocity; p := best position that it has found so far

g := the best position that has been found so far in its neighborhood

U(0,c1) is a random vector uniformly distributed in [0,c1]

⊗ denotes the element-wise multiplication operator

The description of the original PSO can be stated as:

a) Randomly initialize particle positions and velocities

b) While not terminate

   i) For each particle *i* :

      - Evaluate the desired optimization fitness function
      - Compare it with its local best (pBest), update if necessary
      - Compare it with the global best (gBest), update if necessary

   ii) For each particle, update its velocity and position:

$$v_i \longleftarrow v_i + \mathrm{U}(0,\mathrm{c}_1) \otimes (\mathrm{p}_i - \mathrm{x}_i) + \mathrm{U}(0,\mathrm{c}_2) \otimes (\mathrm{p}_i - \mathrm{x}_i)$$

$$x_i \longleftarrow x_i + v_i$$

Note that higher acceleration coefficients result in less stable systems where the velocity has a tendency to explode. This is one disadvantage of this algorithm. Second, limiting the velocity does not necessarily prevent particles from leaving the search space, nor does it help to guarantee convergence. Another is that the

velocity is usually kept within the range ($\mathbf{V}_{max}$). One key of this algorithm is that the position must be usually kept within the range.

2. Inertia weighted PSO

The concept of an inertia weight was developed to better control exploration and exploitation. The aim of inertia weight was to be able to control the exploration mechanism and eliminate the need for $\mathbf{V}_{\mathbf{max}}$. The inclusion of an inertia weight in the PSO algorithm was first published in 1998. The inertia weight was successful in addressing first aim but could not completely eliminate the need of velocity clamping. The inertia weight (w) controls the momentum of the particle by weighting the contribution of the previous velocity. Equations (1) and (2) describe the velocity and position update equations with an inertia weight included.

$$v_i \leftarrow \omega v_i + U(0,c_1) \otimes (p_i - x_i) + U(0,c_2) \otimes (g_i - x_i) \quad (1)$$

$$x_i \leftarrow x_i + v_i \quad (2)$$

This update rule allows for convergence without the use of velocity range ($\mathbf{V}_{\mathbf{max}}$).

Rule-of-thumb settings: ω = 0.7298and $c_1 = c_2$ = 1.49618

3. Constricted coefficients PSO

The constriction coefficient was introduced as an outcome of a theoretical analysis of swarm dynamics (Clerc and Kennedy 2002). Velocities are constricted, with the following change in the velocity update:

$$v_i \leftarrow \chi(v_i + U(0,c_1) \otimes (p_i - x_i) + U(0,c_2) \otimes (g_i - x_i))$$

Convergence is guaranteed under the conditions that

$$c = c_1 + c_2 > 4, and\ \chi = \frac{2}{c+\sqrt{c^2-4c}}$$

4. Logistic-based constricted coefficients PSO

After testing the Original PSO, Inertia weighted PSO and Constricted coefficients PSO algorithm, we found out that the algorithms have some drawback in the term of the converge. For the Original PSO, it considers 100% of Momentum component; the particles will continue their previous motion. In the early stage, the acceleration coefficients should be set sufficiently high, so the swarm could have enough energy to explore the search space. However, the velocity has a tendency to explode in the later stage because of higher acceleration coefficients result in less stable systems. In the other hand, if acceleration coefficients were set low in the beginning, particles move very slow which lead to the converge problem as well. To fix this problem, the Maximum of velocity has been proposed by the author. However, limiting the velocity does not necessarily prevent particles from leaving the search space, nor does it help to guarantee convergence. Because velocity only makes sense when we mean it in the right background and different problem have different scale. So only setting up the safe guard for the velocity still couldn't overcome this drawback.

About Inertia weighted PSO and Constricted coefficients PSO, the weighted coefficient or Constricted coefficient have been added in the original updating process. By setting the coefficient ω greater than one and less than zero, they both

discount the effect of momentum component so that the algorithm could converge given the large enough maximum iteration number set up. Eberhart and Shi also suggested decreasing ω over time (typically from 0.9 to 0.4) and thereby gradually changing from an exploration to exploitation so that the convergence of process could be fast in the later stage of the particles' moving. In our testing study, we found out that the value of parameters in both Inertia weighted PSO and Constricted coefficients PSO should be wisely selected so as to have a decent result. (optimal solution). For both cases, the selection of the parameters is the problem dependent, which could lead to unstable result.

Based on the analysis of different PSO algorithms and testing study, we proposed a novel PSO algorithm (logitPSO), which is variant algorithm of Constricted coefficients PSO with the sum of ϕ_1 and ϕ_2 fixed but their value change over time from a logistic function so that particles are changing dramatically at certain small interval from an exploration to exploitation.

The logit PSO was proposed as following:

- Randomly initialize particle positions and velocities. (population size depends on the search space)
- While not terminate (All particles are closed enough to the globel max or the maximum iteration has been reached)
    - For each particle i :
        - Evaluate the desired optimization fitness function
        - Compare it with its local best (pBest), update if necessary
        - Compare it with the global best (gBest), update if necessary
    - For each particle, update its velocity and position:

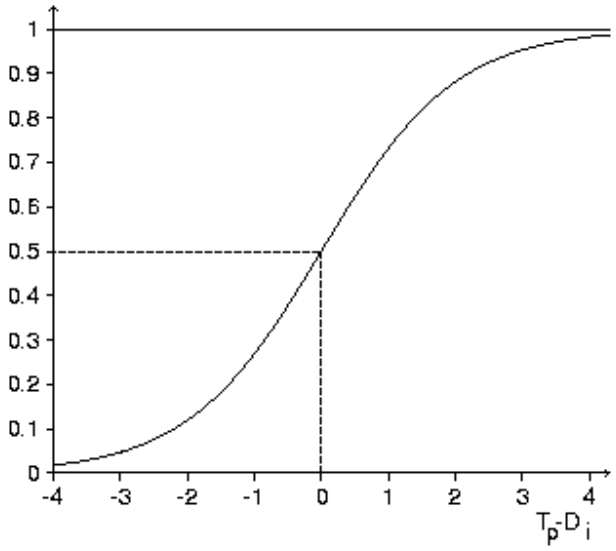
$$\vec{v}_i \leftarrow \chi \cdot \left(\vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{g}_i - \vec{x}_i)\right)$$

The constricted coefficient is defined as follow:

$$\chi = \begin{cases} 1, \phi_1 > \phi_2 \\ \dfrac{2}{\phi + \sqrt{\phi^2 - 4\phi}}, & \phi_1 < \phi_2 \end{cases}$$

Where $\phi = \phi_1 + \phi_2$ and $\phi_2 = \dfrac{\phi}{1 + e^{N/2 - k}}$, and N is the maximum iteration number and k is the iteration number.

Based on the nature of the logistic function which can be seen in the following graph ϕ_2 is relatively small in the first half stage of the whole process, in where the local component is dominating the whole update process and the algorithm is mainly about the exploration. When k is greater than our pre-set switch point, ϕ_2 is increasing dramatically and in this stage, in where the global component is dominating the whole update process and the algorithm is working on the exploitation. By designing the parameters changed over time based on the logistic function of guarantee maximum iteration number and its iteration number will guarantee the convergence of the algorithm.



Testing Study

1. Two dimension SIAM function

Here we test these four algorithms on two-dimensional SIAM function. The range is $[-0.1, 0.1]^2$. It is a multimodel function with 6 local minimums. The optimal function value is -1.864183379308362

First, we used the original PSO to solve it and got the following results:

c_1	c_2	x	f
0.5	**4**	**(0.073989, 0.047033)**	**-1.734773**
4	**0.5**	**(0.072733, 0.046374)**	**-1.734772**
0	**2**	**(0.075191, -0.093183)**	**-1.864183**
2	**0**	**(-0.023386, 0.047773)**	**-1.861672**
2	**2**	**(0.074477, 0.046680)**	**-1.734773**

Table I

From Table I, we see that when $c_1 = 0$ and $c_2 = 2$ the result is the best. In this case, the algorithm puts more 'efforts' to find the global optimum.

Then we use the Inertia weighted PSO to solve the same problem and got the result:

weight	c1=c2	x	f	No. of Iteration
0.7298	**1.49618**	**(0.073584629022878, 0.046124034082284)**	**-1.734773021811118**	**500**
0.5	**1.49618**	**(0.022531759829990, 0.005818958450126)**	**-1.734773021811118**	**500**
0.7	**1.49618**	**(0.027244766510358, -0.088923333425637)**	**-1.864183379308362**	**500**
0.9	**1.49618**	**(0.026328347338718, -0.049650109134200)**	**-1.864183379307904**	**500**

Table II

From Table II, we learned that when weight=0.7, the results are better.

The **constricted coefficients PSO** is employed to solve it again. Results are listed as:

c1	c2	x	f	No. of Iteration
1	**4**	**(0.074587999105150 , -0.092084380747153)**	**-1.864183379308362**	**85**
2	**3**	**(0.073584628703011 , 0.046124034340407)**	**-1.734773021811117**	**58**
3	**2**	**(0.073589218892025 , 0.046145655099420)**	**-1.734771690319168**	**500**
4	**1**	**(0.074876073508515 , 0.019093728765458)**	**-1.864183379245115**	**500**

TableIII

We got Table III without setting V_{max}.

Then we set $V_{max} = 0.001$ and got Table IV.

c1	c2	x	f	No. of Iteration
1	**4**	**(0.046424363190564, 0.043796115527356)**	**-1.383690685967499**	**500**
2	**3**	**(0.047075669674033, -0.042561737834101)**	**-1.767801956648655**	**500**
3	**2**	**(0.065607101979307, 0.016740756647798)**	**-1.516538889228189**	**500**
4	**1**	**(0.073609844387129, 0.045546011869654)**	**-1.733878318501754**	**500**

Table IV

As the end of this section, we use the improved **logitPSO** method and get optimal function value of -1.864183379308356 with optimal point (0.074617237633969, -0.092086037908579) after 500 iterations.

2. Two Dimension Rosenbrock function

Here we test these three algorithms on two-dimensional Rosenbrock function. The range is [-10, 10]^2 . It is a unimodel function with just 1 local and 1 global minimum. The optimal function value is 0 with the value of coordinate (1, 1).
Original PSO:

First, we used the **original PSO** to solve it and got the following results:

C1	C2	x	y	f	Steps
0.001	**0**	**1.848079**	**3.464551**	**0.960840**	**500**
0	**0.001**	**1.055420**	**1.073297**	**0.168034**	**500**
0.001	**0.001**	**0.943059**	**0.891614**	**0.003750**	**500**
0.001	**0.002**	**1.011934**	**1.029707**	**0.003388**	**500**
0.002	**0.001**	**1.114412**	**1.242337**	**0.013108**	**500**

Table V

From Table V, we see that when $c_1 = 0.001$ and the result c2 = 0.002 is the best. However, the convergence rate is very low.

About Inertia weighted PSO:

w	C1	C2	x	y	f	Steps
0.3	**1.49618**	**1.49618**	**1.183961191953628**	**1.402429416456036**	**0.033885984204462**	**500**
0.3*	**1.49618**	**1.49618**	**1.050740930664612**	**1.104049930632629**	**0.002574646364804**	**500**
0.5	**1.49618**	**1.49618**	**2.316519516405876**	**5.368288395064672**	**1.733633993326170**	**500**
0.5*	**1.49618**	**1.49618**	**1.000000002032773**	**1.000000004192520**	**5.744383999612768e-018**	**500**
0.7298	**1.49618**	**1.49618**	**0.999996971969966**	**0.999994058449295**	**1.047999535255821e-011**	**500**
0.7298*	**1.49618**	**1.49618**	**1.000000221024119**	**1.000000443790460**	**4.915517809560085e-014**	**500**
0.9	**1.49618**	**1.49618**	**1.000417116486312**	**1.000833587998260**	**1.740532327878462e-007**	**500**
0.9*	**1.49618**	**1.49618**	**0.855483014525343**	**0.728477376134214**	**0.022023419836644**	**500**

Table VI

Note: * represents the case of no using V_{max}.

From Table VI, we can see that for Rosenbrock function, when $\omega = 0.5$, $c_1 = c_2 = 0.49618$ is the best, and even better than Rule-of-thumb settings: $\omega = 0.7298$ and $c_1 = c_2 = 1.49618$. So, we can say the selection of parameters is problem dependent.

And we also can see that when choosing an appropriate *ω, no using of* V_{max} is better than using of V_{max}.

However, the convergence rates of both using of V_{max} and no using of V_{max} are low.

Constricted coefficients PSO:

C1	C2	x	y	f	Steps
1	**4**	**0.2074**	**0.0393**	**0.6296**	**96**
1*	**4**	**0.3522**	**0.1173**	**0.4242**	**61**
2	**8**	**1.4179**	**2.0114**	**0.1747**	**500**
2*	**8**	**2.4258**	**5.8853**	**2.0330**	**44**
3	**12**	**1.5179**	**2.3042**	**0.2682**	**500**
3*	**12**	**1.3112**	**1.7205**	**0.0970**	**37**

Note: * represents the case of no using V_{max}.

Table VII

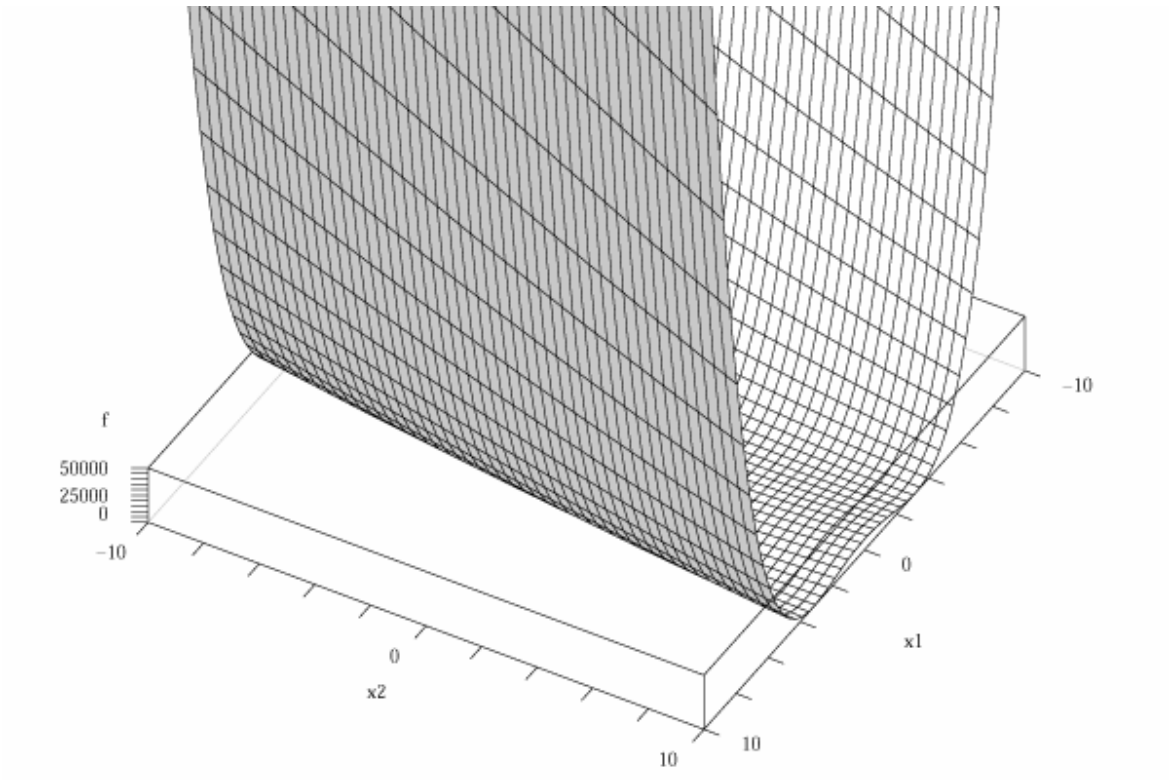
From Table VII, we can see that by using Constricted coefficients PSO algorithm, the convergence rate of no using V_{max} is much higher than that of using V_{max}.

3. High-D Rosonbrock function

We further consider testing all the PSO algorithms on the Generalized Rosenbrock function, which is defined as following:

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

Difficulty level of the testing function is defined by the formula $-\ln(\sigma)$, where σ is the probability of success by randomly choosing a position in the search space.



Rosenbrock Function is represented here on $[-10,10]^2$. There is a barely noticeable global minimum at (1, 1). For the majority of optimization algorithms it is difficult to find the global minimum, and PSO in its initial versions and also our proposed version are no exception when the dimension becomes larger.

Then we checked the difficulty given from the table at Page 57 Particle Swarm Optimization Maurice Clerc, 2005. It shows as following:

Dimension	Difficulty
2	20
5	60
10	120
20	245
30	370

Table 4.2. *Rosenbrock Function. Theoretical difficulty according to the number of dimensions*

We can see that t the evolution of the difficulty according to the dimensionality of the problem increases is almost linear. However, because Difficulty level of the testing function is defined by the formula $-\ln(\sigma)$, The true difficulty thus increases exponentially.

On the 5-dimensiton Rosenbrock testing function:

	dimension	Value of Object function	Position attained the minimum	converge
oPSO	**5D**	**0.00027224**	**(1.0003,1.0002,1.0005,1.0002, 0.9990)**	**500***
wPSO	**5D**	**1.53E-14**	**(1.0000 1.0000 1.0000 1.0000 1.0000**	**500***
ccPSO	**5D**	**1.1515**	**(0.8090 0.6571 0.4367 0.1899 0.0217)**	**500***
logitPSO	**5D**	**6.57E-05**	**(1.0009 1.0018 1.0035 1.0070 1.0141)**	**440**

* Indicates the maximum iteration number has been reached.

We found out that Inertia weighted PSO has the best result among all the algorithms, followed by our logitPSO, Constricted coefficients PSO and the original PSO.

On the 5-dimensiton Rosenbrock testing function:

	dimension	Value of Object function	converge
oPSO	**10D**	**1.786600**	**500***
wPSO	**10D**	**0.047223**	**500***
ccPSO	**10D**	**0.467570**	**500***
logitPSO	**10D**	**0.00098987**	**459**

* Indicates the maximum iteration number has been reached.

We found out that our logitPSO has the best result among all the algorithms, followed by our Inertia weighted PSO, logitPSO, Constricted coefficients PSO and the original PSO.

Conclusion

We have studies the particle swarm optimization algorithm and other improved versions. Based on our analysis and understanding of PSO, we developed a novel algorithm which tried to overcome the problem of the stable velocity letting the parameters changed by logistic function over the time. In our study, we found out that selection of the parameters is problem-dependent, which is crucial to the

behavior of the algorithm. We had different result by setting up different parameter. Secondly, we tried to develop a novel PSO algorithm, However, Results are hard to compare due to the stochastic result. Each time the program gave us the different result for the large dimension problem, which indicates that the population size is not large enough. We learnt that high dimension optimization problem is hard to solve. Because when search space is huge, PSO loses its power given a limit population size. In our testing, ccPSO works better for SIAM function wPSO works better for Rosenbrock function.

References:

1. Particles Swarm Optimization, James Kennedy and Russel Eberthart. 1995
2. Good Parameters Particle Swarm Optimization, Magnus Erik and Hvass Pedersn. 2010
3. wiki page about PSO: http://en.wikipedia.org/wiki/Particle_swarm_optimization
4. Particles Swarm Optimization, Maurice Clerc. 2005
5. Particles Swarm Optimization, Aleksandar Lazinica. 2009
6. Simplifying Particle Swarm Optimization, Magnus Erik Hvass Pedersen, Andrew John Chipperfield. 2009
7. A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, Suraj Pandey, LinlinWu, Siddeswara Mayura Guru, Rajkumar Buyya.