Hybrid particle swarm algorithm for solving nonlinear constraint optimization problems

BINGQIN QIAO, XIAOMING CHANG
Computers and Software College
Taiyuan University of Technology
Department of Economic Information, SXFTC
Taiyuan 030024
CHINA

MINGWEI CUI, KUI YAO
Institute of Science
PLA University of Science and Technology
Nanjing 211101
CHINA
Correspondence author: flyyaok@sina.com

Abstract: Based on the combination of the particle swarm algorithm and multiplier penalty function method for the constraint conditions, this paper proposes an improved hybrid particle swarm optimization algorithm which is used to solve nonlinear constraint optimization problems. The algorithm converts nonlinear constraint function into no-constraints nonlinear problems by constructing the multiplier penalty function to use the multiplier penalty function values as the fitness of particles. Under the constraint conditions to make the particle swarm track the best particle and fly to the global best, this paper is to redefine p-best which is the particles position last iteration and g-best which refers to the best position of the group last iteration. And, by redefining p-best and g-best, the particle can avoid tracking the p-best and the g-best whose fitness are excellent but violate constraints to great extent, so that particle swarm would finally converge to the optimum position in the feasible domain. Numerical experiments show that the new algorithm is correct and effective.

Key–Words: Particle swarm optimization; Multiplier method; Multiplier penalty function; Nonlinear constraint optimization; Nonlinear constraint; Global optimization.

1 Introduction

The constrained optimization problem is a branch of the optimization problems, its mathematical model can be formulated as follows:

$$\begin{cases} \text{Minimize} & f(x) \\ \text{Subject to} & g_i(x) \geq 0, i = 1, ...m; \\ & h_{\mathrm{j}}(x) = 0, j = 1, ...l. \end{cases} \quad (1)$$

where $f(x)$,

$$g_i(x) \geq 0 (i = 1, ...m)$$

and

$$h_j(x) = 0 (j = 1, ...l)$$

are functions defined in R^n.

If there is a nonlinear function contained in $f(x)$ or $g_i(x)$ or $h_j(x)$, (1) is called nonlinear constraint optimization problems. Let $S \subseteq R^n$ denote the searching space of the optimization problem. The constrained optimization problem is looking for the minimum point of target function $f(x)$ in N-dimension Euclidean space in the feasible region

$$F = \left\{ x \in R^N \mid \begin{array}{l} g_{\mathrm{i}}(x) \geq 0, i = 1, \cdots, m; \\ h_{\mathrm{j}}(x) = 0, j = 1, \cdots, l \end{array} \right\},$$

which is composed of inequality constraints $g_i(x) \geq 0$ and equality constraint $h_j(x) = 0$. In the constrained extreme value problem, since the decision variables $x = (x_1, x_2, \cdots x_n) \in F \subseteq S \subseteq R^n$ smooth point (stationary point) of the objective function in unconstrained might not be in the feasible region, so generally unconstrained extreme value conditions cannot be used to solve constrain problem.

At present, there are two types of main methods to solve the constrained optimization problem: certainty method and randomness method. Certainty methods often require that target function and constraint function are continuous and differentiable. Sequential quadratic programming method, Penalty function method and Lagrange multiplier method all belong to certainty method. The convergence speeds of these algorithms are faster, but they are easy to go into the local minimum value. So in generally they are suitable for solving the optimization problem of smaller dimension. Random method has not the strict requirements for the objective function and constraint function. Genetic algorithm and Differential evolution algorithm have been widely used in solving the constrained optimization problem [5].

Particle Swarm Optimization (briefly, PSO) was firstly proposed by Kennedy and Eberhart in 1995,

and it is an evolutionary algorithm based on iterations. Since this algorithm has some advantages such as less parameter, easy to realize, fast convergence speed and better optimization ability, it has been paid widely attention and been studied after it was put forward. However, how to use it to solve nonlinear programming problem with constraints is still at the primary stages [5]. Parsopoulos et al. [6] brought the penalty function into particle swarm algorithm to solve the optimization problem. During the initial and iterative process, since those particles did not meet the constraint condition, Gao et al. [4] used the external point method to replace the original particles of new particles under constraint conditions. The role of the external point method is to take advantage of penalty function to change the constrained optimization into unconstrained optimization. When the penalty factor tends to limit, penalty function becomes more and more ill-conditioned and brings great difficulties for function calculation. In particular, to determine appropriate penalty coefficients becomes very difficult, that is associated with the optimization problems itself. Hu and Eberhart et al. [7] gave a method that all particles within the group were initialized to the feasible region, and then the particles beyond the feasible region did not participate in the competition with the best particle in the following iterative process. The disadvantage of this method is the difficulty in initializing particles when the feasible region is very small. Pulido and Coello [11] gave a clever strategy of selecting the optimal particle, which did not need to initialize particles in the feasible region, but by comparing the position of the particles in the feasible region with the particles in non-feasible region, it can make the whole group gradually converge to the feasible region. However, particles in non-feasible region could also serve as attractors; this method would lead to some computation wasted in non-feasible region. Ma et al. [5] proposed a strategy that two good positions in feasible region worked as attractors, in which each particle used the global best "p_g" as well as using the nearest "p_i" in the feasible region to guide the search, and this "p_i" could belong to other particles. However, this algorithm requested a particle of the initial population was located in the feasible region.

This paper introduces the multiplier method to handle the constraints by combining particle swarm optimization algorithm with multiplier method, adding a multiplier penalty function to the objective function and constructing augmented Lagrange function as the fitness function, and then the minimum value problem of replaced the objective function in (1) by the augmented Lagrange function. Penalty factor does not tend to infinity, so that multiplier method would be able to obtain the optimal solution of the constrained problem. And ill-condition will not appear in the penalty function method. Multiplier method is a way to solve the constrained optimization problems by revising multiplier vector and penalty factor continuously, which makes the degree of constraint violations of solution become smaller and smaller and gradually approach to the best point of the original constrained problem. According to this feature of the constrained optimization, in this paper, each particle of p_g follows the optimal position of the particle swarm groups in the previous iteration rather than group's historical optimum position, and p_i is the particle individual position in the previous iteration rather than individual historical optimum position, because the latter often shows better fitness when constraint violations extent is greater. The improved hybrid algorithm combines the advantages of random optimization ability of particle swarm optimization with the superiority of the multiplier method. Numerical experiments show that the new algorithm has good effectiveness and feasibility.

2 Improved Particle Swarm Optimization

Search of the particle swarm algorithm depends on the N-particles flying in the D-dimensional search space (N-particles have no size and no weight). Each particle represents a solution in search space, the flight position of the i-th particle $(1 \leq i \leq N)$ is represented as $x_i = (x_{i,1}, x_{i,2}, \cdots x_{i,D})$, the flight speed is expressed as $v_i = (v_{i,1}, v_{i,2}, \cdots v_{i,D})$, and its flight is influenced by its own speed v_i, the best location of individual history $p_i = (p_{i,1}, p_{i,2}, \cdots p_{i,D})$ and the particle groups optimal position $p_g = (p_{g,1}, p_{g,2}, \cdots p_{g,D})$. In the standard particle swarm algorithm, for each particle, its dimension d changes according to (2), (3):

$$\begin{aligned} v_{i,d}(t+1) = w v_{i,d}(t) + c_1 r_1 [p_{i,d} - x_{i,d}(t)] \\ + c_2 r_2 [p_{g,d} - x_{i,d}(t)] \end{aligned} \quad (2)$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1), d = 1, 2, \cdots, D \quad (3)$$

where, t is the number of iterations, w is inertia weight, c_1 and c_2 are learning factors, r_1 and r_2 is a random number between (0,1).

The extreme points of the particles within the search range are called the global optimal solution. In this condition, optimization objective has no constraint or constraints do not work. In the standard particle swarm algorithm, for each particle, to compare its fitness and the best position p_i which has gone through, if the former is better, it can be used to update p_i; then to compare all p_i and p_g, if the p_i is better than p_g then update p_g. The speed of the particle

is affected by the individual historical extreme value points and the global extreme value points, so that all of the particles gradually converge to the best point of the group, which is the function extreme value point. However, for the objective function with constraints, most of the extreme value point of the function and the unconstrained extreme value point of the function is not the same point, and the unconstrained value of the function may be superior to the constraint value of the function. During solving the constrained optimization problem, according to the standard particle swarm algorithm, the particles will fly in the whole searching range in early iteration because of the randomness of the initial particles. And the particles are influenced by individual history extreme value point and global extreme value point, which will lead the particles to fly toward extreme value point within the scope of the search in the iteration process, and the particles will deviate from the constraint domain and hard to converge to the extreme value point of constraint domain. This paper presents a new method to guide the flight of particles, in (2), for each particle, its position of the previous generation is taken as the particles individual history extreme points p_i of the current generation, and the particle groups optimal position of the previous generation is taken as p_g of current generation in each iteration. That is, in flight, the particle is only affected by the individual particles position and excellent particle's position in groups of previous generation, but not affected by the particles position in earlier generation. Because the particle of the earlier generation violates the constraint in a larger degree, such particle is not a global optimum but a local optimum even if its fitness is very excellent. In particular, the learning factor c_1 and c_2 take the same value generally, and their values between 0-4, in most cases, $c_1 = c_2 = 2$. However, in the experiments of this paper, set c_1 and c_2 between 0-1, which make the particle attracted less by the individual extreme value and group extreme value.

Inertia weight w determines the influential degree of the particle's previous velocity on the current speed, and the right choice can make the particle have a balanced exploration and development capabilities. This paper adopts the exponent decrease inertia weight in document Ref [9].

$$w = (w_{start} - w_{end} - d_1)e^{\frac{1}{1+d_2 t/t_{\max}}} \qquad (4)$$

where, t_{max} is the maximum iterations number, t is the current iterations number, w_{start} is the initial inertia weight, w_{end} is the inertia weight when the particle swarm evolutes to the maximum number of iterations, d_1 and d_2 are controlling factors to control w between w_{start} and w_{end} . In the experiment of this paper, set w_{start} =0.95, w_{end} =0.4, d_1 =0.2, d_2 =7.

3 Constraint Processing Technology

Processing the constraints is the key when using particle swarm optimization solves constrained optimization problems. Traditional constraints handling methods are to construct the penalty function or augment Lagrange function, which can transform the constrained problems into unconstrained problems. The basic idea of the penalty function method is: at the feasible point, penalty function value is equal to the original objective function value, and at the unfeasible point, the penalty function value is equal to a great value. But the inherent defect of the penalty function method is that only when penalty factor tends to infinity can the objective functions optimal solution obtain, which requires constructing a suitable penalty function based on the actual situation. If a penalty factor is too large, which may cause calculation difficulties in penalty function minimization, and if its too small, the calculation result may not be the most optimum target, or convergence speed is too small.

In order to overcome the shortcomings of the penalty function, in 1969, Hestenes and Powell proposed the multiplier method respectively. This method does not require the penalty factor which tends to infinity. As long as its value is great enough, the augmented Lagrange function can be minimized to get the minimum values of the objective function.

To solve the constrained optimization problem described in (1), the augmented Lagrange function is defined as follows:

$$\begin{aligned}\phi(x,\omega,\upsilon,\sigma) = f(x) \\ + \tfrac{1}{2\sigma}\sum_{i=1}^{m}\{[\max(0,\omega_i - \sigma g_i(x))]^2 - \omega_i^2\} \\ - \sum_{j=1}^{l}\upsilon_j h_j(x) + \tfrac{\sigma}{2}\sum_{j=1}^{l}(h_j(x))^2\end{aligned} \qquad (5)$$

where, σ is a sufficient large parameter, ω_i and σ_i are the multipliers. The multiplier iteration formula is (t is the number of iterations):

$$\begin{cases}\omega_i(t+1) = \max(0, \omega_i(t) - \sigma g_i(x(t))), \\ \upsilon_j(t+1) = \upsilon_j(t) - \sigma h_j(x(t)), \\ i = 1,\cdots,m, j = 1,\cdots,l.\end{cases} \qquad (6)$$

There are four parts in the right hand side of (5): the first is original objective function, the second is multiplier penalty term of inequality constraint, the third is multiplier term of equality constraint and the

fourth is penalty term of equality constraint. If problems are only concerned with equality constraints, augmented Lagrange function in (5) would only include the first, third and fourth item; if problems are only concerned with inequality constraints, augmented Lagrange function in (5) would only include the first and second term.

If the boundary of the feasible region is near the boundary of the search space, some particles may violate the constraints on the feasible region boundary in the iterative process. Therefore, the particles in the search space boundary are reassigned by the following formula [4]:

$x_{id}(t)$ can be defined as

$$\begin{cases} \overline{x}_d(t) + r(x_d^l - \overline{x}_d(t)), \text{if } \ x_{id}(t) < x_d^l \\ \overline{x}_d(t) + r(x_d^u - \overline{x}_d(t)), \text{if } \ x_{id}(t) > x_d^u \end{cases} \tag{7}$$

where,$r \in U[0,1]$,

$$\overline{x}_d(t) = (\sum_{i=1}^{N} x_{id}(t))/N$$

denotes the mid-value of $x_{id}(t)$ on the dimension d, N is the size of the particle swarm.

4 Hybrid Particle Swarm Algorithm

In (1), the basic process of the hybrid particle swarm algorithm which uses multiplier method to handle the constraints is defined as follows:

Step 1. Initialize randomly N-particles in the search range. Define the initial multiplier vector ω and υ , the penalty factor $\sigma > 0$, constant magnification factor $\alpha > 1$ and the parameter $\beta \in (0,1)$.

Step 2. Calculate the fitness of each particle: $\phi(x, \omega, \upsilon, \sigma, \zeta)$, according to (5).

Step . Store the current particle position into the particles individual best value p_i , and store the optimal p_i into the p_g .

Step 4. If

$$\frac{\left\|h(p_g^t)\right\|}{\left\|h(p_g^{t-1})\right\|} \geq \beta$$

or

$$\frac{\left\|g(p_g^t)\right\|}{\left\|g(p_g^{t-1})\right\|} \geq \beta$$

(it is used to measure the convergence speed), set

$$\sigma = \alpha\sigma,$$

go to step 5; otherwise, go to step 5.

Step 5. Update the multiplier vector ω and υ of the next generation of globally optimal particle according to (6).

Step 6. Update particles according to (2) and (3), and deal with cross-border particles according to (7).

Step 7. Repeat steps 2 to 7 until get a preset maximum number of iterations or get a sufficiently good fitness value.

5 Numerical Experiments

5.1 Test Functions

(1) Function f_1

$$\min f_1(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

s.t.

$$(x_1 - 5)^2 + (x_2 - 5)^2 \geq 100;$$

$$-(x_1 - 6)^2 - (x_2 - 5)^2 \geq -82.81,$$

and

$$13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$$

The optimal solution (20 times average) obtained in this article:

$$x^* : 14.09505035874353 \ \ 1.50564302677460$$

and

$$f_1(x^*) : \ \ -6961.690026419559$$

In the particles iterative process, the f_1 function value evolution curve is shown in Fig.1 (due to the initial particles are randomly generated in the search range, the optimal function evolution curve is quite different for each experiment, so the function evolution diagram comes from a single experiment, the diagram below is same).

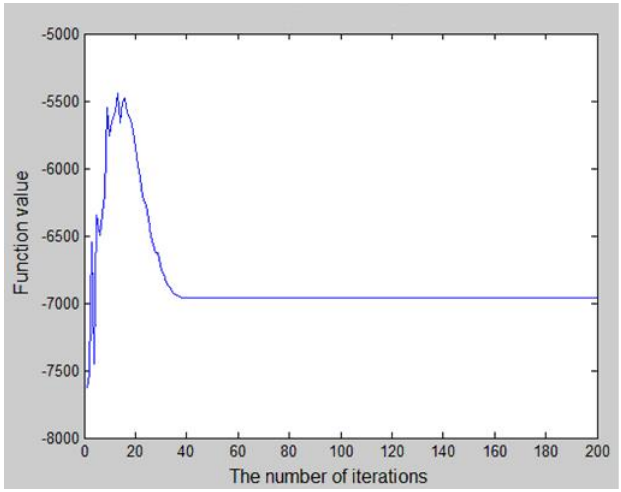

Fig.1 the evolution curve of the function value

(2) Function f_2

$$\min f_2(x) = x_1$$

s.t.

$$g_1(x) = \frac{1}{4}x_1 + \frac{1}{2}x_2 - \frac{1}{16}x_1^2 - \frac{1}{16}x_2^2 \leq 1,$$

$$g_2(x) = \frac{1}{14}x_1^2 + \frac{1}{14}x_2^2 - \frac{3}{7}x_1 - \frac{3}{7}x_2 \leq -1,$$

$$1 \leq x_1 \leq 5.5,$$

$$1 \leq x_2 \leq 5.5.$$

The optimal solution (20 times average) obtained in this article:

$$x^* : 1.17712434446771,\ \ 2.17712434446770$$

$$f_2(x^*) : 1.17712434446771$$

In the particles iterative process, the f_2 function value evolution curve is shown in Fig.2.

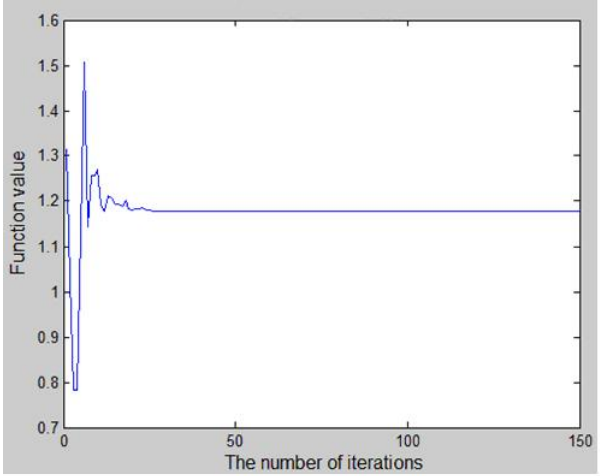

Fig.2 the evolution curve of the f_2 function value

(3) Function f_3

$$\min f_3(x) = 0.5x_1x_2^{-1} - x_1 - 5x_2^{-1}$$

s.t.

$$g_1(x) = -0.01x_2x_3^{-1} - 0.01x_2 - 0.0005x_1x_3 \geq 1,$$

$$70 \leq x_1 \leq 150, 1 \leq x_2 \leq 30,$$

and

$$0.5 \leq x_3 \leq 21$$

The optimal solution (20 times average) obtained in this article:

$$x^* : \begin{matrix} 150.000704216473,\ \ 27.539275040623, \\ 0.622706026129, \end{matrix}$$

$$f_3(x^*) : -147.3463377232924$$

In the particles iterative process, the f_3 function value evolution curve is shown in Fig.3.

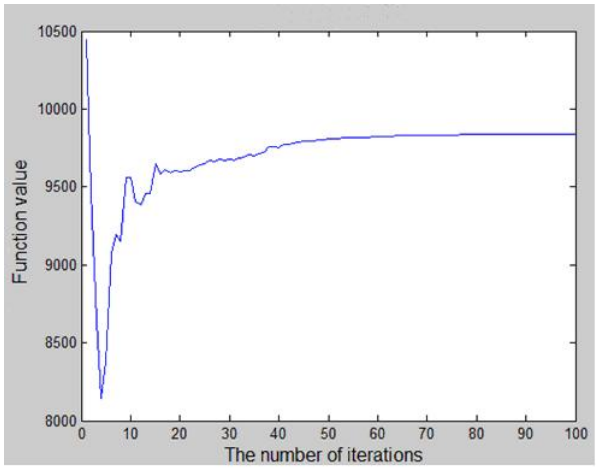

Fig.3 the evolution curve of the f_3 function value

(4) Function f_4

$$\min f_4(x) = 5.3578x_3^2 + 0.8357x_1x_5 + 37.2392x_1,$$

s.t. $g_1(x) =$

$$0.00002584x_3x_5 - 0.00006663x_2x_5 - 0.0000734x_1x_4 \leq 1,$$

$g_2(x) =$

$$0.000853007x_2x_5 - 0.00009395x_1x_4 - 0.00033085x_3x_5 \leq 1,$$

$g_3(x) =$

$$1330.3294x_2^{-1}x_5^{-1} - 0.42x_1x_5^{-1} - 0.30586x_2^{-1}x_3^2x_5^{-1} \leq 1,$$

$g_4(x) =$

$$0.00024186x_2x_5 + 0.00010159x_1x_2 + 0.00007379x_3^2 \leq 1,$$

$g_5(x) =$

$$2275.1327x_3^{-1}x_5^{-1} - 0.2668x_1x_5^{-1} - 0.40584x_4x_5^{-1} \leq 1,$$

$g_6(x) =$

$$0.00029955x_3x_5 + 0.00007992x_1x_3 + 0.00012157x_3x_4 \leq 1,$$

$$78.0 \leq x_1 \leq 102.0,$$

$$33.0 \leq x_2 \leq 45.0,$$

$$27.0 \leq x_i \leq 45.0 (i = 3, 4, 5)$$

The optimal solution (20 times average) obtained in this article:

$$x^* : \begin{matrix} 78.04210611686725,\ \ 39.98691692228350, \\ 28.82163615787091,\ \ 42.59148530225040, \\ 41.16955092852577 \end{matrix}$$

$$f_4(x^*) : 10056.67707387459$$

In the particles iterative process, the f_4 function value evolution curve is shown in Fig.4.

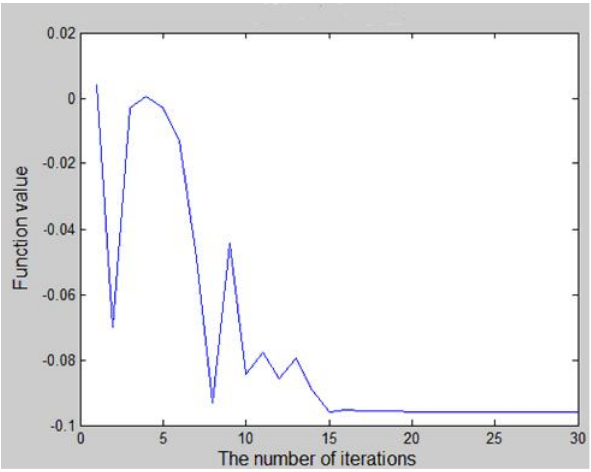

Fig.4 the evolution curve of the f_4 function value

(5) Function f_5

$$\max f_5(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$$

s.t.

$$g_1(x) = x_1^2 - x_2 + 1 \le 0,$$
$$g_2(x) = 1 - x_1 + (x_2-4)^2 \le 0,$$
$$0 \le x_1 \le 10, 0 \le x_2 \le 10$$

The optimal solution (20 times average) obtained in this article:

$$x^* : 1.22797483152742,\quad 4.24536144358884$$

$$f_5(x^*) : 0.09582503370020$$

In the particles iterative process, the f_5 function value evolution curve is shown in Fig.5.

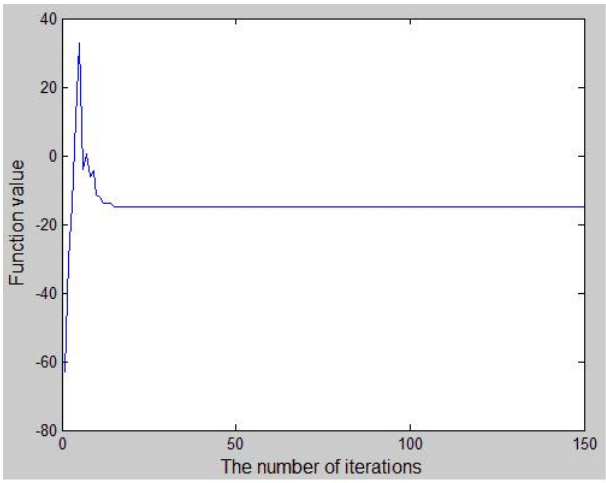

Fig.5 the evolution curve of the f_5 function value

(6) Function f_6

$$\min f_6(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

s.t.

$$g_1(x) = -2x_1 - 2x_2 - x_{10} - x_{11} + 10 \ge 0,$$
$$g_2(x) = -2x_1 - 2x_3 - x_{10} - x_{12} + 10 \ge 0,$$
$$g_3(x) = -2x_2 - 2x_3 - x_{11} - x_{12} + 10 \ge 0,$$
$$g_4(x) = 8x_1 - x_{10} \ge 0,$$
$$g_5(x) = 8x_2 - x_{11} \ge 0,$$
$$g_6(x) = 8x_3 - x_{12} \ge 0,$$
$$g_7(x) = 2x_4 + x_5 - x_{10} \ge 0,$$
$$g_8(x) = 2x_6 + x_7 - x_{11} \ge 0,$$
$$g_9(x) = 2x_8 + x_9 - x_{12} \ge 0,$$
$$0 \le x_i \le 1 (i = 1, 2, \cdots 10),$$
$$0 \le x_i \le 100 (i = 10, 11, 12),$$
$$0 \le x_{13} \le 1.$$

The optimal solution (20 times average) obtained in this article:

$$\begin{array}{rll} & 0.99336221512764, & 0.99264796634399 \\ & 0.99865332896608, & 0.99820446255649 \\ & 0.98913679087585, & 0.99554085574522 \\ x^* : & 0.97933187185900, & 0.99712837737829 \\ & 0.98766628324966, & 2.98281047858041 \\ & 2.95910435327020, & 2.97708566917939 \\ & 0.97973504534171 & \end{array}$$

$$f_6(x^*) : -14.76663551960394$$

In the particles iterative process, the f_6 function value evolution curve is shown in Fig.6.

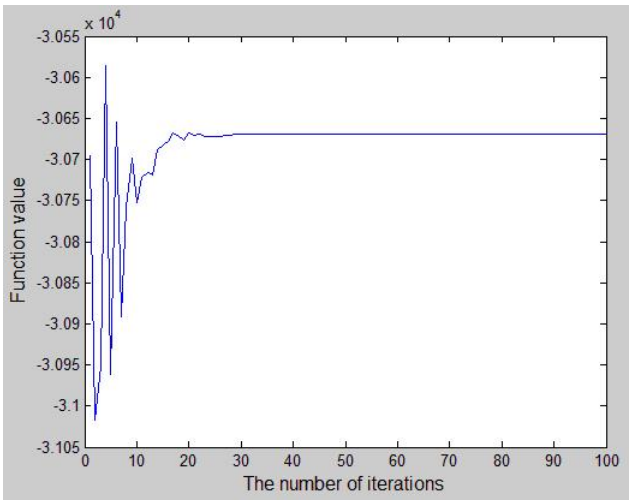

Fig.6 the evolution curve of the f_6 function value

(7) Function f_7

$$\min f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

s.t.

$$\begin{aligned}
g_1(x) &= 92 - 85.334407 - 0.0056858x_2x_5 \\
&\quad -0.0006262x_1x_4 + 0.0022053x_3x_5 \geq 0 \\
g_2(x) &= 85.334407 + 0.0056858x_2x_5 \\
&\quad + 0.0006262x_1x_4 - 0.0022053x_3x_5 \geq 0 \\
g_3(x) &= -80.51249 - 0.0071317x_2x_5 \\
&\quad - 0.0029955x_1x_2 - 0.0021813x_3^2 + 110 \geq 0 \\
g_4(x) &= 80.51249 + 0.0071317x_2x_5 \\
&\quad + 0.0029955x_1x_2 + 0.0021813x_3^2 - 90 \geq 0 \\
g_5(x) &= 25 - 9.300961 - 0.0047062x_3x_5 \\
&\quad - 0.0012547x_1x_3 - 0.0019085x_3x_4 \geq 0 \\
g_6(x) &= -20 + 9.300961 + 0.0047062x_3x_5 \\
&\quad + 0.0012547x_1x_3 + 0.0019085x_3x_4 \geq 0
\end{aligned}$$

with

$$78 \leq x_1 \leq 102,$$

$$33 \leq x_2 \leq 45,$$

$$27 \leq x_i \leq 45 (i = 3, 4, 5)$$

The optimal solution (20 times average) obtained in this article:

$$x^* : \begin{matrix} 78.00000000003647 & 33.00000583946766 \\ 29.98645470860217 & 44.99967834081389 \\ 36.77005736212263 \end{matrix}$$

$$f_7(x^*) : -30657.22185557836$$

In the particles iterative process, the f_7 function value evolution curve is shown in Fig.7.

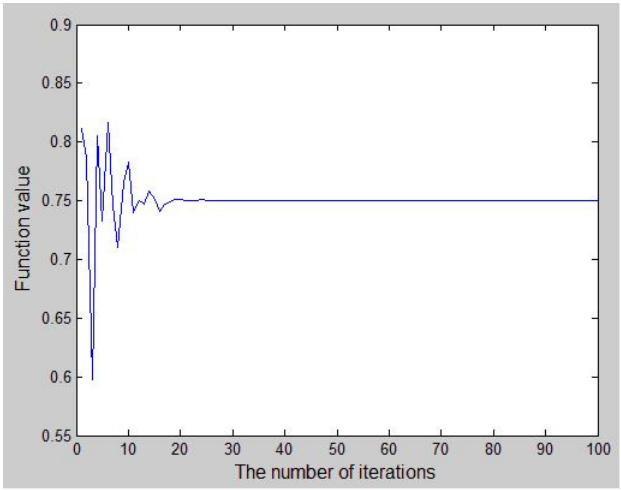

Fig.7 the evolution curve of the f_7 function value

(8) Function f_8

$$\min f(x) = x_1^2 + (x_2 - 1)^2$$

s.t.

$$h_1(x) = x_2 - x_1^2 = 0$$

$$-1 \leq x_1 \leq 1,$$

$$-1 \leq x_2 \leq 1$$

The optimal solution (20 times average) obtained in this article:

$$x^* : \begin{matrix} 0.70710677997036 & 0.49999999806761 \\ -0.70710677997036 & 0.49999999806761 \end{matrix}$$

$$f_8(x^*) : 0.75000000021244$$

In the particles iterative process, the f_8 function value evolution curve is shown in Fig.8.

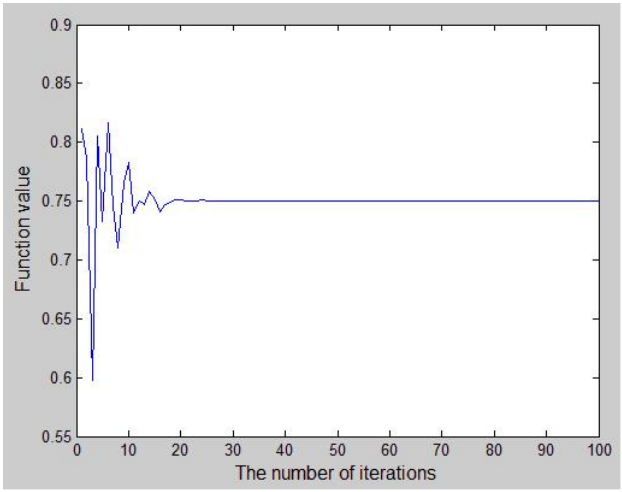

Fig.8 the evolution curve of the f_8 function value

5.2 Experiment Analysis

Numerical experiments are processing in MATLAB 7.0.1. In the calculation, set learning factors c_1 = c_2 = 0.7 for f_6 function, c_1 = c_2 = 0.6 for f_7 function, c_1 = c_2 = 1.0 for the remaining functions. Population size: f_1 ,f_2 ,f_3 ,f_4 ,f_5 ,f_6 ,f_7 ,f_8 for 50 particles, f_4 f_5 for 20 particles. Experiment iteration time will change with the scope and complexity of the problem. Set multiplier vector $\omega = 1, \upsilon = 1$, and β = 0.7 to measure the speed of convergence, and set constant magnification factor α = 2, and the initial penalty factor σ settings: function f_4 , f_6 and f_7 are set to 200, the remaining functions are set as 20. For each test function, the algorithms are run 20 times, taking the optimal value, worst value, mean, mean square error. The results are shown in Table 1.

F	Optimal value
f_1	-6961.81387558016
f_2	1.17712434446771
f_3	-147.6669538913704
f_4	9761.567601684035
f_5	-0.09582504140719
f_6	-14.99905931197651
f_7	-30668.74234793204
f_8	0.74999999910799
Worst value Mean	Mean
-6960.57538397414	-6961.690026419559
1.17712434446771	1.17712434446771
-145.7188592406307	-147.3463377232924
10605.74154900046	10056.67707387459
-0.09582499088190	-0.09582503370020
-14.43006360512358	-14.76663551960394
-30625.85294512852	-30657.22185557836
0.75000000347674	0.75000000021244
square error	Iterations
0.38119967310441	200
0	150
0.67731369342482	300
286.8661296805540	100
1.302403541283220e-008	30
0.18495823750212	150
15.71720094738370	100
9.708100746526781e-010	100

Table 1 The operation results of the algorithm

Comparing a new algorithm with the Ref. [1,2,4,8,10], deterministic global optimization method were used for solving constrained optimization problems in Ref.[1,8]; in Ref.[2], the method of bi-objective fitness was used to deal with the constraints; in Ref.[4], external point method was used to handle the constraints, then the problem is solved by the adaptive velocity particle swarm optimization algorithm. The comparison of new algorithm and other papers is shown in Table 2-9.

The functions in this paper are common test functions for constraint optimization problems, in which f_5 is maximization problem that the objective function may be transformed into minimization problem by negation, and f_8 is the equality constrained optimization problem. It is difficult to achieve global optimization when using evolutionary algorithm for these test functions. It can be seen from Table 2-9 that the new algorithm on these constrained optimization problems basically achieved the optimal value, especially for f_4 function. The optimal solution found by the new algorithm is better than the solutions in other papers. Momentum particle swarm algorithm in the Ref. [5] will cause difficulty in the problem solving because the particles follow the best value of their own history and the best value of group. In many cases at the early stage, the particles fitness which violates the constraints more is better than the particles fitness which violates the constraints less, so that the particles may fly out of the feasible region.

Let IVN denote the Independent variable number, the following Table.2 -9 shows the results comparing the present paper with Ref [1-10].

Table 2. Compared results for f_1

F	IVN	Reference	Optimal value
f_1	2	[2]	-6961.81388
f_1	2	[4]	-6961.8138665
f_1	2	[10]	-6961.81388
f_1	2	This paper	-6961.81387558016

Table 3. Compared results for f_2

F	IVN	Reference	Optimal value
f_2	2	[4]	1.177643
f_2	2	[8]	1.177124327
f_2	2	This paper	1.17712434446771

Table 4. Compared results for f_3

F	IVN	Reference	Optimal value
f_3	3	[1]	-83.249728406
f_3	3	[4]	-147.6667
f_3	3	[8]	-83.249728406
f_3	3	This paper	-147.6669538913704

Table 5. Compared results for f_4

F	IVN	Reference	Optimal value
f_4	5	[1]	10122.493176362
f_4	5	[4]	10122.49323
f_4	5	[8]	10122.38112168
f_4	5	This paper	9761.567601684035

Table 6. Compared results for f_5

F	IVN	Reference	Optimal value
f_5	2	[2]	-0.095825
f_5	2	[4]	-0.0958250
f_5	2	[10]	-0.095825
f_5	2	This paper	-0.09582504140719

Table 7. Compared results for f_6

F	IVN	Reference	Optimal value
f_6	13	[2]	-15
f_6	13	[4]	-14.9999999
f_6	13	[10]	-15
f_6	13	This paper	-14.99905931197651

Table 8. Compared results for f_7

F	IVN	Reference	Optimal value
f_7	5	[2]	-30655.539
f_7	5	[4]	-30665.53677
f_7	5	[10]	-30655.539
f_7	5	This paper	-30668.74234793204

Table 9. Compared results for f_8

F	IVN	Reference	Optimal value
f_8	2	[2]	0.749
f_8	2	[4]	0.7500000
f_8	2	[10]	0.749
f_8	2	This paper	0.74999999910799

6 Conclusion

This paper proposes a hybrid particle swarm optimization algorithm based on the multiplier penalty function to solve constrained optimization problems. During the search process, this method sets the particles position of the previous generation as the optimum of particle individual history and the optimum of group of the previous generation as the optimum of group, which makes each flight of the particle be affected only by the flight of the previous generation particle. This is conducive to trace changes of constraint violations extent, and makes all of the particles gradually approaching the feasible region, and eventually finds the optimal solution in the feasible region. The particle swarm optimization algorithm has no requirement for the optimization function. The numerical experiments show that the new algorithm has the ability to find an optimal value under constraint condition. The parameter adjustment of the new algorithm is more difficult for the optimization problem of different levels of complexity, especially the difference in the number of iterations and the number of particles is large, but the solution speed of the particle swarm algorithm is affected by the number of iterations and the size of the particle swarm.

Acknowledgements: The research was supported by NSFC (No.11071282), and Supported by the Open Project Program of Key Laboratory of Intelligent Computing and Information Processing of Ministry of Education, Xiangtan University, China (No.2011ICIP07). Many thanks for Prof. Sheng Baohuai for his helpful suggestions.

References:

[1] P. P. Shen, K. C. Zhang, Global optimization of signomial geometric programming using linear relaxation, *Appl. Math. Comput.* No.150, 2004, pp. 99-114.

[2] H. Y. Lu, W. Q. Chen, Self-adaptive velocity particle swarm optimization for solving constrained optimization problems, *Journal of Global Optimization.* Vol.41, No.3, 2008, pp. 427-445.

[3] C. Gong, Z. L. Wang, *Proficient in MATLAB optimization.* Beijing: publishing house of electronics industry, 2010.

[4] Y. L. Gao, H. R. Li, Hybrid Particle Swarm Algorithm of Nonlinear Constraint Optimization Problems, *Mathematica Numerica Sinica.* Vol.32, No.2, 2010, pp. 135-146.

[5] R. X. Ma, Y. Liu , Z. Qin, X. Wang, Momentum Particle Swarm Optimizer for Constrained Optimization, *Journal of System Simulation.* Vol.22, No.11, 2010, pp. 2485-2488.

[6] K. E. Parsopoulos, M. N. Vrahatis, Particle Swarm Optimization Method for Constrained Optimization Problems, *Intelligent Technologies-Theory and Applications: New Trends in Intelligent Technologies, USA: IEEE.* No.12, 2002, pp. 214-220.

[7] X. H. Hu, R. C. Eberhart, Y. H. Shi, Engineering optimization with particle swarm, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, USA: IEEE.* 2003, pp. 53-57.

[8] P. P. Shen, H. W. Jiao, A new rectangle branch-and-pruning approach for generalized geometric programming, *Applied Mathematics and Computation.* No.183, 2006, pp. 1027-1038.

[9] Y. H. Shi, R. C. Eberhart, Empiricalstudy of particle swarm optimization, *Proc of Congress on Computational Intelligence, Washington DC, USA.* 1999, pp. 1945-1950.

[10] H. Y. Lu, W. Q. Chen, Dynamic-objective particle swarm optimization for solving constrained optimization problem, *J. Comb. Optim.* No.12, 2006, pp. 409-419.

[11] G. T. Pulido, C. A. C. Coello, A constraint-handling mechanism for particle swarm optimization, *IEEE 2004 Congress on Evolutionary Computation.USA: IEEE.* 2004, pp. 1396-1403.