International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.7 (2014), pp.97-108
http://dx.doi.org/10.14257/ijmue.2014.9.7.09

# Investigation of Modified PSO Algorithm used in the Solution of Problems with Constraints and its Application

Kang Lv[1] and Yuanyuan Ma[2]

[1]*Henan Institute of Education, Zhengzhou, Henan, China*
[2]*Henan Normal University, Xinxiang, Henan, China*
[1]*24295376@163.com*, [2]*hnxxmyy@sina.com*

***Abstract***

*With the development of computer technology, more and more intelligent algorithms in the solution of different problems in many aspects of society are extensively used. Calculation of the minimum values of the function optimization and finding the optimal solution of combinatorial optimization problems in certain space are the two typical problems. In the fields of science and engineering, many optimization problems are constrained with different conditions. Due to the existence of constraints, the optimization problems are more difficult than the unconstrained optimization problem. Therefore, research on how to make the objective function find the optimal solution in the feasible region is of great significance As an important intelligent algorithm, particle swarm optimization (PSO) is widely used in the calculation of the optimal solution of the constraint problems. But due to the disadvantages of the PSO, its search efficiency is quite low when the particle is close to the optimal value. On the other side, it is easy to search the local optimal solution but difficult to get the global optimal solution. So, it is necessary to modify the algorithm to improve the performance. In the paper, we modify the algorithm and propose a modified algorithm (M-PSO). With the simulation, the results show the validity of the algorithm.*

***Keywords:*** *intelligent algorithm, M-PSO, constraints, application.*

## 1. Introduction

With the development of computer technology, more and more intelligent algorithms in the solution of different problems in many aspects of society are extensively used. At the same time, supercomputer with rapid development provides the hardware support to realize an intelligent algorithm. Intelligent algorithms are often used to solve the general optimization problem, which can be divided into two kinds: (1) calculate the minimum values of the function optimization, and (2) find the optimal solution of the combinatorial optimization problems in a solution space. There are many optimization algorithms, the classical algorithms include linear programming, dynamic programming, and so on. Improved local search algorithm consists of hill climbing method and steepest descent method. Optimization algorithm can also be divided as guided search algorithm and system dynamic evolution method. The former one includes simulated annealing [1-3], genetic algorithm [4-6], tabu search [7-8], while the latter one includes neural network, chaos search algorithm.

Particle swarm optimization (PSO) is proposed by Kennedy and Eberhart as a new bionic optimization algorithm. It originates from the study of the foraging behavior of the birds. It has fast convergence performance on some issues and requires fewer parameters [9-11]. It has the disadvantage of slow convergence speed at later evolution, may not find the optimal solution for complex problems, and not high accuracy. In order to avoid the defects, many studies have been done to improve the performance [12-20].

ISSN: 1975-0080 IJMUE

(1) Hybrid PSO (HPSO) [21]

In the hybrid model, the new particle swarm produced by the iteration would select based on the fitness value. Position and velocity of half particles with high fitness degree would substitute that of half particles with lower fitness degree. The individual extreme would be maintained unchanged. HPSO improves the convergence speed and maintain the global convergence ability. It has better performance than the PSO in the majority of function optimization problems. However, it is somewhat limited in solving complex optimization problem with high dimensional, nonlinear, and large number of local extreme values.

(2) Stochastic PSO (SPSO) [22]

The basic idea is to use the stop evolution particles to improve the global search ability. Velocity would lose the memory by removing previous velocity item. It weakens the ability of global search while it also makes at least one particle in each generation locate at historical best position. New particles are generated randomly to replace stopping particles. This method improves the global searching ability. The algorithm also improves that the PSO algorithm cannot guarantee the global convergence.

(3) The synergy PSO algorithm [23-24]

The basic idea of synergy PSO algorithm is that $N$ independent particle swarms search along $D$ directions in space. The specific approach is: select partition factor $N$ and particle number $M$ in particle swarm. Input vector with $D$ dimensions (velocity and position vector) into $N$ particle swarms. The first $D \bmod N$ particle swarms has velocity and position vector of particles with $D / N$ dimensions. The latter $K - (D \bmod N)$ particle swarms has velocity and position vector of particles with $D / N$ dimensions. In each iteration, the N particle swarms update the status independently with no shared information.

In the calculation of fitness value, the particle with optimum position in each particle swarm would splice together as a $D$ dimension vector, which is taken into the calculation of the fitness value. Synergistic PSO algorithm can jump out of the local extreme to achieve higher convergence precision

(4) Hybrid PSO [25]

A particle is given a crossover probability, which is given by the user himself. In each iteration, some amount of particles would be selected and put into a pool based on the hybrid probability. Random hybrid would be happened in the pool among the particles and produce the same number of progeny particles, which are used to substitute the parent particles to keep the particles number constant. Progeny particle's position is determined by the parent particle's position. The crossover may happen in different subgroup and the same subgroup. The hybrid algorithm has a higher convergence speed than that of PSO. Search accuracy is relatively high for some nonlinear optimization problems. But due to more referred parameters to adjust, number of iteration steps will increase. So there are certain requirements for the user’s experience

In this paper, we propose a modified PSO (M-PSO) to avoid the disadvantages of the PSO. The main contribution is the proposition of the new algorithm. The remainder of the paper is shown as the following: Description of the original PSO are listed in Section 2. Modified PSO is shown in Section 3. Application is shown in section 4. And the conclusion is described in Section 5.

 

# 2. PSO

## 2.1. Standard PSO Algorithm

Particle swarm optimization (PSO), which is a kind of evolutionary algorithm, is also based on the group. Each potential solution of optimization problems can be seen as a particle in the searching space. Each particle has a speed to determine their flight direction and distance, and then the particles will follow the current optimal particle to search in the solution space. The PSO algorithm is initialized as a group of random particles and the particles will fly at a certain speed in the space. The optimal solution will be found according to the iteration. In each iteration, the particle will update the value by following two extreme values. One is the optimal solutions found by particle itself, and the other is the optimal solution found in the whole space.

Make some assumption: $n$ represents the dimensions of space; $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ means the current position of particle $i$; $P_i = (p_{i1}, p_{i2}, ..., p_{im})$ represents the optimal position found by particle $i$; $g$ represents the optimal particle in the space; $v_i = (v_{i1}, v_{i2}, ..., v_{im})$ is the velocity of particle $i$. The velocity and position would be updated by the equation (1).

$$\begin{cases} v_i^{k+1} = \omega_i^k \times v_i^k + c_1 \times rand\ (\bullet) \times (p_i^k - x_i^k) + c_2 \times Rand\ (\bullet) \times (P_g - x_i^k) \\ x_i^{k+1} = x_i^k + v_i^{k+1} \end{cases} \quad (1)$$

Where, $k$ is the number of iteration steps; $c_1, c_2$ is the learning factor; $rand\ (\bullet)$, $Rand\ (\bullet)$ are the random number in interval $[0,1]$; $\omega_i^k$ is the inertia weight.

Another important kinds of PSO algorithm is particle swarm optimization algorithm with convergence factors (CPSO), and the velocity will update with the following:

$$v_i^{k+1} = \alpha \times (v_i^k + c_1 \times rand\ (\bullet) \times (p_i^k - x_i^k) + c_2 \times Rand\ (\bullet) \times (P_g - x_i^k)) \quad (2)$$

Here, we call the $\alpha$ as the convergence factor, and

$$\alpha = \frac{2}{\left|2 - \varphi + \sqrt{\varphi^2 - 4\varphi}\right|} \quad (3)$$

$$\varphi = c_1 + c_2, \qquad \varphi > 4 \quad (4)$$

In some condition, the CPSO has a better performance than that of PSO.

## 2.2. Discrete Particle Swarm Optimization Algorithm (DPSO)

(1) Optimization design Model for discrete variables

Optimization design model for nonlinear constraints discrete variables can be summarized as follows:

$$\begin{cases} \min \quad f(x) \qquad X = \{x_1, x_2, ..., x_n\} \qquad x_i \in D_i \\ \quad Di = \{d_{i,1}, d_{i,2}, ..., d_{i,m}\} \qquad i = 1,2,..., n \\ s.t. \quad g_k(x) \le 0 \qquad k = 1,2,..., N_c \end{cases} \quad (5)$$

Where, $x_i$: the discrete variable $i$;

$D_i$ is the solutions set of discrete design variables $i$;

$d_{i,1}; d_{i,m}$ : the upper and lower limits of $i$; $m$ is the number of discrete solutions, while $m$ may be different for each discrete variable;

$N_c$ : the number of constraints function;

$D_i$ : discrete design variable;

$D_i$ : Objective function;

$D_i$ : performance constraint function.

(2) Penalty function method

Penalty function method is widely used in the genetic algorithm and the traditional gradient algorithm to solve the constraint problem. But there are fewer examples of using the penalty function method to solve the discrete variables. Here, the PSO method is combined with a penalty function method to solve the nonlinear constrained discrete variable problem. That is to say the point in the space would be eliminated when it is not meeting the demand of constraint conditions.

For the expansion function based on PSO algorithm, the form would be described as the following:

$$F(X) = f(X) + s\phi(X) + rG(X) \tag{6}$$

Where, $s$ means the discrete penalty factor and $r$ is the constraints penalty factor.

Here, the expansion function $F(X)$ is composed of an objective function $f(X)$, discrete penalty function $s\phi(X)$ and constraint penalty function $rG(X)$. $f(X)$ would be replaced by $F(X)$, and optimization problem with constraint discrete variables would be transformed into problem with unconstrained continuous variables.

The penalty function in the following would be adopted by equation (6).

$$\Phi(x) = \sum_{i=1}^{n} \frac{1}{2}\left[\sin \frac{2\pi[x_i^* - 0.25(d_{i,j+1} + 3d_{i,j})]}{d_{i,j+1} - d_{i,j}} + 1\right] \tag{7}$$

Where, $d_{i,j}$ and $d_{i,j+1}$ are the discrete variable; $x_i^*$ is the continuous variable between $d_{i,j}$ and $d_{i,j+1}$.

The exterior penalty function would be adopted as the penalty function for constraint functions.

$$G(X) = \sum_{k=1}^{N_c} \max[0, g_k(x)]^2 \tag{8}$$

Due to the introduction of discrete penalty function, expanding function $F(X)$ becomes highly continuous function with non-convexity. If solving the problem with traditional gradient optimization method, global optimal function, even local optimal solution, would not be obtained. The reason is the gradient method is not suitable for solving continuous non convex problem. PSO is very suitable for solving continuous problem with non-convexity to obtain the global optimum. Then, the optimization design problem with nonlinear constraints discrete variables is transformed to the following:

 

$$\begin{cases} \min \quad F(X^*) \\ \\ d_{i,1} \le x_i^* \le d_{i,m} \qquad i = 1,2,\ldots, n \end{cases} \tag{9}$$

### 2.3. Flow Chart of PSO

①The swarm parameter would be initialized and size of the space would be determined. Position and velocity of each particle would be given.

② Corresponding calculation of fitness value for each particle.

③ For each particle, compare the fitness value and individual extreme value. If the fitness value is bigger than individual extreme value, the fitness value would be adopted to replace the individual extreme value.

④ Compare the fitness value with global extreme value. If the fitness value is bigger than global extreme value, the global extreme value would be replaced.

⑤ With corresponding formula, velocity and position of each particle would be updated.

⑥ The calculation would be finished if the results meet the demand of end conditions (error is adaptable, or gets the maximum cycle number), otherwise repeat step 2.

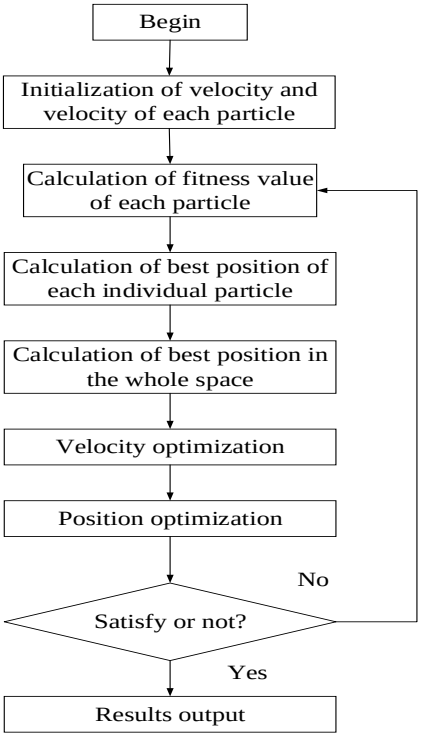


**Figure 1. Flow Chart of PSO**

## 3. Modified PSO

### 3.1. Trust Region combined Inertia Weight

In the search process, if a particle finds an optimal position, other particles would quickly move closer to it. In some problem with complex objective function, particles may not find

the global optimal solution and just get the local optimal solution. In order to avoid premature convergence in the local search, trust region algorithm combined with PSO is used in the calculation to adjust the inertia weight. When the particles began to search, inertia weight would be adjusted to strengthen the global search capability. When the function is decreasing at a certain value, the inertia weight should be adjusted to strengthen the local search ability.

$F$ is the objective function and some parameters would be defined: (1) $F_i^t$ is the objective function of particle $i$ after $t$ times of iteration; (2) $F_i^{t-1}$ s the objective function of particle $i$ after $t-1$ times of iteration; (3) $F_g^{t-1}$ is the global optimal objective function value of all the particles after $t-1$ times of iteration. Then the actual reduction of particle $i$ is $F_i^t - F_i^{t-1}$ and $F_i^t - F_g^{t-1}$ is the predicted reduction. If we define $r_k$ as the following:

$$r_k = \frac{f_i^t - f_i^{t-1}}{f_i^t - f_g^{t-1}} \tag{10}$$

Then, we can judge whether the inertia weight is suitable based on the approximation degree of $r_k$ to 1. That is to say that we can adjust the weight with the inertia described as the following:

(1) Give some constants as $\eta$ , $\eta_1$ , $\eta_2 \in (0.1)$ , and $\eta < \eta_1 < \eta_2$ . Generally, $\eta$ is approximately to 0, and $\eta_2$ is approximately to 1.

(2) If $r_k > \frac{3}{4}$, we can think that there is the good approximation between the actual reduction and predicted reduction. This means objective function value of the particle has a good decline in the original direction and longer search step would make the better performance. Then, in order to improve the search ability in a new region, we should enlarge the radius of the trust region of inertia weight $\omega$ . It is $\omega_i^{t+1} = \min \{2\omega_i^k, \omega_{\max}\}$.

(3) If $\eta < r_k < \frac{1}{4}$, we can think that there is the bad approximation between the actual reduction and predicted reduction. We should strengthen the local search ability. Decrease the radius of the trust region of inertia weight $\omega$ . That is $\omega_i^{t+1} = \max \{2\omega_i^k, \omega_{\min}\}$

(4) If $\frac{1}{4} < r_k < \frac{3}{4}$, we can think that there is the good approximation between the actual reduction and predicted reduction. Inertia weight algorithm has a balance in the global search ability and local search ability. Then we would keep the original radius constant. That is $\omega_i^{t+1} = \omega_i^k$

Based on the inertias described above, iterative formula of particle swarm optimization algorithm can be updated as the following:

$$v_{ij}^{t+1} = \text{int}( \omega_i^t \times (v_{ij}^t + c_1 \times rand\ (\bullet) \times (p_{ij}^t - x_{ij}^t) + c_2 \times Rand\ (\bullet) \times (p_{gj} - x_{ij}^t)) \tag{11}$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \tag{12}$$

 

$$\omega_i^{t+1} = \begin{cases} \omega_{\max}, & r_k > \frac{3}{4} \quad and \quad 2\omega_i^t \geq \omega_{\max} \\ 2\omega_i^k, & r_k > \frac{3}{4} \quad and \quad 2\omega_i^t \leq \omega_{\max} \\ \omega_{\min}, & r_k < \frac{1}{4} \quad and \quad \frac{1}{2}\omega_i^t \leq \omega_{\min} \\ \frac{1}{2}\omega_i^k, & r_k < \frac{1}{4} \quad and \quad \frac{1}{2}\omega_i^t \geq \omega_{\min} \\ \omega_i^k, & \frac{1}{4} \leq r_k \leq \frac{3}{4} \end{cases} \tag{13}$$

$$r_k = \frac{f_i^t - f_i^{t+1}}{f_i^t - f_g^{t+1}} \tag{14}$$

Where, $f_i^t$ is the objective function of particle $i$ after $t$ times iteration; $f_i^{t+1}$ is the objective function of particle $i$ after $t+1$ times iteration; $f_g^{t+1}$ is the global optimal objective function of all the particles after $t+1$ times iteration. $f_i^t - f_i^{t+1}$ is the actual reduction of particle $i$ after $t$ times iteration; $f_i^t - f_g^{t+1}$ is the predicted reduction of particle $i$ after $t$ times iteration;

### 3.2. Processing of Constraints

$$r_k = \frac{f_i^t - f_i^{t-1}}{f_i^t - f_g^{t-1}} \tag{15}$$

Based on the traditional constraint optimization method and combined with the equation (15), constraint optimization method is modified. When it exceeds the feasible region, it should be processed with the following.

(1)When $r_k > \frac{1}{4}$, $x_i(t+1)$ exceeds the feasible domain. Due to the $f(x_i(t+1)) < f(x_i(t))$, the direction is a descent direction, and there is high probability of smaller value than $f(x_i(t))$ on the vector. So in order not to waste particles exercise, particle would not return to the last generation. In this condition, we will select an arbitrary position between the last generation position and the current position. And the particle would move by the following path:

$$x_i(t+2) = x_i(t) + (1-\lambda)x_i(t+1) \tag{16}$$

Where, $\lambda \in (0,1)$. The particle will move until it returns to the feasible region

(2) When $r_k < \frac{1}{4}$ and $x_i(t+1)$ exceeds the feasible domain, we think the particles have worse declining performance. In this condition, there is lower possible to have a lower value than $f(x_i(t))$, then the particle would return to the last generation position.

The calculation method is described as the following:

(1) If $r_k > \frac{1}{4}$, $x_c = x_i(t+1)$, then $x_c = \lambda x_i(t) + (1-\lambda)x_c$, $\lambda \in (0,1)$

(2) If $r_k < \frac{1}{4}$, $x_i(t+2) = x_i(t)$

Where, $r_k = \frac{f_i^{\,t} - f_i^{\,t+1}}{f_i^{\,t} - f_g^{\,t+1}}$

### 3.3. Flow Chart of Modified Algorithm

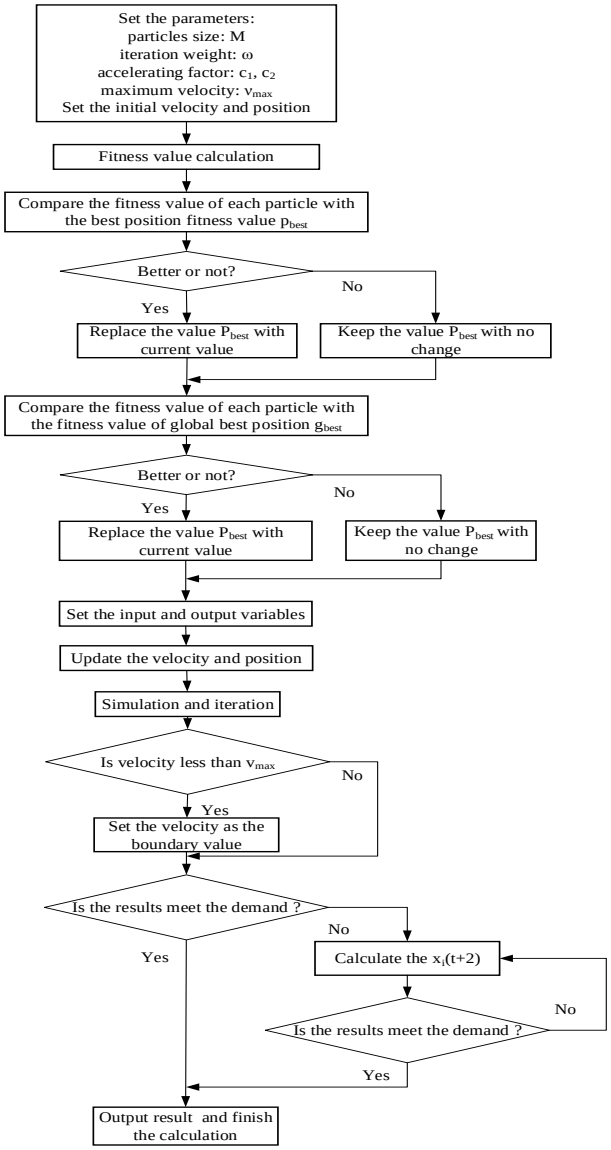


**Figure 2. Flow Chart of Modified PSO**

## 4. Application

We will take the optimization design of pressure vessel as the example. The calculation results would be compared with the CARLOS [26] and HE [27] methods.

 

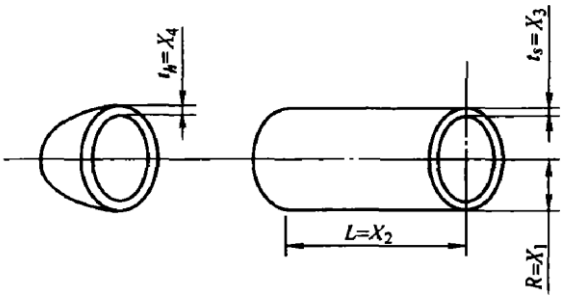


**Figure 3. Sketch Map of the Pressure Vessel**

As shown in Figure 3, the problem can be described as the design variables of $x_1(R)$, $x_2(L)$, $x_3(t_s)$, $x_4(t_h)$. The aim of the problem is design the vessel with least material. $x_1$, $x_2$ are the continuous variable, and $x_3$, $x_4$ are discrete variable.

Firstly, we should establish the mathematics model as the following:

Calculate the $X(x_1, x_2, x_3, x_4)$ and get the $\min \ f(X)$.

$$f(X) = 0.622 * x_1 * x_2 * x_3 + 1.778 * x_1^2 * x_4 + 3.166 * x_2 * x_3^2 + 19.84 * x_1 * x_1^3 \tag{17}$$

And the constraints are described as the following:

$$\begin{cases} 10 \le x_1 \le 200 \\ 10 \le x_1 \le 200 \\ 0.05 \le x_3 \le 7 \\ 0.05 \le x_1 \le 7 \end{cases} \tag{18}$$

$$\begin{cases} g_1(x) = \dfrac{0.0193 \ x_1}{x_3} - 1 \le 0 \\ g_2(x) = \dfrac{0.00954 \ x_1}{x_4} - 1 \le 0 \\ g_3(x) = \dfrac{x_2}{240} - 1 \le 0 \\ g_4(x) = \dfrac{1296000 \ - \frac{4}{3}\pi x_1^3}{\pi x_1^2 x_2} - 1 \le 0 \end{cases} \tag{19}$$

With the experiments, the initial parameters would be adopted as the table 1 shows.

**Table 1. Initial Parameters**

| Items | Value |
|---|---|
| The initial constraint penalty factor | 6000 |
| The initial discrete constraint penalty factor | 130 |
| Modified coefficient | 1.002 |
| Maximum times of iteration | 150 |
| Number of the particles | 50 |
| Maximum inertia coefficient | 1.4 |
| Minimum inertia coefficient | 0.3 |

**Table 2. Results Comparison**

| Items | KANNAN | CARLOS | Modified PSO |
|---|---|---|---|
| Radius of the vessel | 58.29 | 40.32 | 42.23 |
| Length of the vessel | 43.36 | 200 | 258.822 |
| Thickness of the vessel | 1.125 | 0.8125 | 0.934 |
| Thickness of the hemispherical head wall | 0.625 | 0.4375 | 0.573 |
| Constraint $g_1$ | 0 | -0.04 | -0.012 |
| Constraint $g_2$ | -1.110 | -0.120 | -0.112 |
| Constraint $g_3$ | -0.82 | -0.17 | -0.0.38 |
| Constraint $g_4$ | -1.11 | 0.07 | 0.001 |
| Vessel weight $f(x)$ | 7198.2 | 6288.7445 | 6938.323 |

With random optimization for 10 times, we can get the results in the calculation as shown in Table 2. This means the modified PSO algorithm is effective.

## 5. Conclusion

With the development of computer technology, more and more extensive application of intelligent algorithms in the solution of different problems in many aspects of society. PSO is an effective algorithm to solve the constraint problems. However, it has also disadvantages in the calculation, so many papers have been published to improve the performance, such as Hybrid PSO (HPSO), stochastic PSO (SPSO), the synergy PSO algorithm, and Discrete particle swarm optimization algorithm (DPSO).

In the paper, we propose a modified PSO. In order to avoid premature convergence in the local search, trust region algorithm combined with PSO is used in the calculation to adjust the inertia weight. When the particles began to search, inertia weight would be adjusted to strengthen the global search capability. When the function is decreasing at a certain value, the inertia weight should be adjusted to strengthen the local search ability. Constraint optimization method is modified. When it exceeds the feasible region, some methods would be used to solve the conditions.

Comparison of the application shows the validity of the modified PSO.

## References

[1] G. Xiutang, X. Jin, X. Jianhua and P. Linqiang, "A simple simulated annealing algorithm for the maximum clique problem", Information Sciences, vol. 177, no. 22, **(2007)**, pp. 5064-5071.
[2] S. S. M. Hossein, M. N. Hassan, L. Gholamhossein, M. M. Mahmoud, S. G. Kambiz, M. Mohammad and M. Y. Hoon, "Experimental and numerical investigation of an adaptive simulated annealing technique in optimization of warm tube hydroforming", Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 226, no. 11, **(2012)**, pp. 1869-1879.
[3] S. Mercedeh, J. Ali, A. Saba and M. Negar, "ParSA: Parallel simulated annealing placement algorithm for multi-core systems", CADS 2012 - 16th CSI International Symposium on Computer Architecture and Digital Systems, **(2012)**, pp. 19-24.
[4] Y. Qiu-Hui, Y. Zhi-Sheng, F. Zi-Liang and H. Mei, "Scheduling arrival aircrafts on multiple runways based on an improved genetic algorithm", Journal of Sichuan University (Engineering Science Edition), vol. 38, no. 2, **(2006)**, pp. 141-145.
[5] Y. Tai-Shan, "Adaptive genetic algorithm simulating human reproduction mode and its application in multi-peak function optimization", 2009 International Workshop on Intelligent Systems and Applications, ISA 2009, **(2009)**.

 

[6] L. Zhiyong, L. Houfu, C. Youwen and A. Sallam, "A hybrid strategy based on niche genetic algorithm and Tabu search and its convergence property", Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010, **(2010)**, pp. 328-332.
[7] C. Yu-Wen, L. Zhi-Qiang and X. Li-Feng, "K-means clustering-tabu search algorithm for collaborative logistics network resource demands design and cost analysis", Journal of Shanghai Jiaotong University, vol. 43, no. 9, **(2009)**, pp. 1407-1411+1416.
[8] Y. Ning, L. Ping and L. Mingsen, "An angle-based crossover tabu search for vehicle routing problem", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), **(2007)**, pp. 4682:1215-1222.
[9] Kennedy, R. Eberhart, "Particle swarm optimization", IEEE international Conference on Neural Networks, **(1995)**, pp. 1942-1948.
[10] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization", Proceedings of the 1999 congress on Evolutionary. Computation, **(1999)**, pp.1945- 1950.
[11] R. Eberhart and Y. Shi, "Particle swarm optimization: development applications and resources", Proceedings of the 2001 Congress on Evolutionary Computation, **(2001)**, pp. 81- 86.
[12] Y. Shi and R. Eberhart, "A modified particle swarm optimizer", IEEE International Conference on Evolutionary Computation, **(1998)**, pp. 69- 73.
[13] Chatt Erjee A, Siarry P. "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization", Computers and Operations Research, vol. 33, **(2006)**, pp. 859- 871.
[14] R. Eberhart and Y. Shi, "Comparing inertia weights and Constriction factors in particle swarm optimization", Proceedings of 2000 Congress on Evolutionary Computation, **(2000)**, pp. 84- 88.
[15] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization", Proceedings of the 2001 Congress on Evolutionary Computation, **(2001)**, pp. 101- 106.
[16] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm", Information Processing Letters, vol. 93, **(2005)**, pp. 255- 261.
[17] S. He, Q. H. Wu, J. Y. Wen, J. R. Saunders and R. C. Paton, "A particle swarm optimizer with passive congregation", Biosystems, vol. 78, **(2004)**, pp. 135-147.
[18] B. Liu, L. Wang, Y. H. Jin, F. Tang and D. X. Huang, "Improve particle swarm optimization combined with Chaos", Chaos, Solitons and Fractacs, vol. 25, **(2005)**, pp. 1261- 1271.
[19] M. Clerc and J. Kennedy, "The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space", IEEE Trans Evolut Comput, vol. 6, no. 1, **(2002)**, pp. 58- 73.
[20] K. Yasuda and N. Iwasaki, "Adaptive particle swarm optimization", IEEE International Conference on Systems, Man and Cybernetics, **(2003)**, pp. 1554- 1559.
[21] P. J. Angeline, "Using Selection to Improve Particle Swarm Optimization", IEEE International Conference on Evolutionary Computation **(1998)**.
[22] F. Solis and R. Wets, "Minimization by random search techniques", Mathematics of Operations Research, 6 vol. 1, **(1981)**, pp. 19-30.
[23] F. Van den Bergh and A. P. Engelbrecht, "Training Product Unit Networks Using Cooperative Particle Swarm Optimization", Proc. of the third Genetic and Evolutionary Computation Conference, (GECCO), **(2001)**.
[24] F. Van den Bergh and A. P. Engelbrecht, "Effect of Swarm Size on Cooperative Particle Swarm Optimization", Proc of the GEOOC, **(2001)**.
[25] M. Lovbjerg, T. K. Rasmussen and T. Krink, "Hybrid Particle Swarm Optimization with Breeding and Subpopulations", Proc of Genetic and Evolutionary Computation Conference, **(2001)**.
[26] B. K. Kannan and S. N. Kramer, "An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design", Journal of Mechanical Design Trans, ASME, vol. 116, **(1994)**, pp. 318-320.
[27] A. Caelos, "Use of self-adaptive penalty approach for engineering optimization problems", Computer in Industry, vol. 41, **(2000)**, pp. 113-127.

## Authors

**Kang Lv,** He received his Bachelor's degree in Computer Science and Technology Specialty (2004) and Master of Science in Computer Engineering Technology (2008). Now He is the lecturer of computer Department, Henan Institute of Education of China. His current research interests include different aspects of Network engineering and Data mining.

**Yuanyuan Ma,** She received her Bachelor's degree in Computer Science and Technology Specialty (2004) and Master of Computer Software and theory (2007). Now she is the lecturer of computer and Information Engineering Department, Henan Normal University of China. Her current research interests include different aspects of Artificial Intelligence and Granular Computer.