RESEARCH ARTICLE OPEN ACCESS

# Hybrid Algorithm of Adaptive Inertia Weight Particle Swarm and Simulated Annealing

XiaoHua. Meng[1], YanFei. Lin[2], DaSheng Qin[3]

[1,2,3]Department of Computer Science and Technology, Jinan University, Guangzhou, China

----------------------------------------************************----------------------------------

**Abstract:**

In order to address the weakness of particle swarm optimization's tendency to easily fall into local optimum in solving large scale combinational optimization problem, considering the balance that inertia can control between local search ability and global search ability, the paper proposed an improved hybrid particle swarm optimization algorithm (PSO) by adopting the self-adaptive inertia weight model and local search strategy of simulated annealing algorithm. Not only increases the variety of particles according to their distance to global optimum, but also enhances the local search ability of the algorithm. The Traveling Salesman Problem (TSP) is adopted to validate the efficiency of the proposed algorithm. By comparing with inertia weight linear decreasing particle swarm optimization, adaptive inertia weight particle swarm optimization and simulated annealing (SA) algorithm, experiments demonstrate that our method has a more promising results, proves it a more efficient modified algorithm.

*Keywords* **—Particle Swarm Optimization algorithm, large scale combination optimization problem, inertia weight, simulated annealing (SA).**

----------------------------------------************************----------------------------------

## I. INTRODUCTION

In terms of NP problem, the problem scale is often large and it is hard to gain the desired optimal solution via accurate algorithm in a reasonable time. However, this problem can be solved well by selecting and applying a proper intelligent algorithm. Generally speaking, intelligent algorithm cannot guarantee that the optimal solution can be obtained, but it can gain the approximately optimal solution in a reasonable time. By weighing the advantages and disadvantages in the game of calculation time and answer quality, intelligent algorithm is a better choice to solve such large-scale combinational optimization problem when compared with accurate algorithm.

Proposed by Doctor Kennedy and Doctor Eberhart[1] in 1995,particle swarm optimization algorithm(PSO algorithm) is an intelligent optimization algorithm. At the beginning, PSO algorithm will generate a group of particles at random (i.e. random solutions) and initial movement velocities of particles. Then it will approach the optimal solution through continuous iteration, and the final result will converge into the optimal solution. As for particle swarm optimization, all particles in the process of particle swarm evolution use the same inertia weight. In this way, the difference among the particles will reduce gradually and the particle velocity will drop. Finally it will tend to be 0.The particle will wander near the optimal solution, but optimal solution of the problem cannot be found accurately. Only a local optimal solution can be gained. If the particle swarm velocity becomes 0,the particle swarm cannot evolve anymore. In another word, it cannot seek optimal solution of the problem by jumping out from the local optimal solution. Therefore, faced with complex problems, PSO algorithm has weak solving ability, and can hardly gain a more accurate optimal solution during the later period.

By directing at the defect of PSO algorithm to fall into local optimum easily when solving complex problems, this paper improves the algorithm. Firstly, different pairs of particles are classified according to the distance between the particles and global optimal solution, and different inertia weights are set for particles of different types, so as to increase the diversity of particles. Secondly, by aiming at the weakness of PSO algorithm in local searching ability, this paper adds the local searching strategy of simulated annealing algorithm into this algorithm. When the algorithm cannot find a more optimal solution after searching for several times, the neighborhood of individual optimal solution in the particles will be searched, thus local searching ability of the algorithm can be enhanced. Finally, performance of the algorithm is verified through experiment. A comparison is made between this algorithm and linear decreasing inertia weight particle swarm optimization algorithm and simulated annealing algorithm(SA).The result shows that the algorithm proposed in this paper has a better effect when solving the travelling salesman problem;it is an improved algorithm with relatively high efficiency.

## II. PSO ALGORITHM

### 1) Basic PSO algorithm

In PSO algorithm, the state of particle at time t in the n-dimensional space can be expressed as $\mathbf{X_i = (X_{i1}, X_{i2}, \ldots, X_{in})}$, and velocity of the particle is $\mathbf{V_i = (V_{i1}, V_{i2}, \ldots, V_{in})}$ . The optimal position experienced by the individual particle is recorded as pbest, and the optimal position experienced by the particle swarm is recorded as **gbest**. Then the velocity and position of the particle at time **t+1** can be updated according to the following formula:

$\mathbf{V_i(t+1) = \omega V_i(t) + c_1 rand(\ \ )(pbest -}$
**Xit+c2randgbest−Xit** (1)

$\mathbf{X_i(t+1) = X_i(t) + V_i(t+1)}$ (2)

In the formula, $\boldsymbol{\omega}$ represents the inertia weight; $\mathbf{c_1}$ and $\mathbf{c_2}$ indicate the acceleration constants. Therefore, the algorithm flow of basic PSO algorithm can be described as the following steps:

Step1: Set the inertia weight and acceleration constants of the algorithm and initialize a group of particles and velocities. The optimal position **pbest** experienced by each individual particle is set as the initial position, and the optimal position **gbest** experienced by the particle swarm is the optimal value among all **pbest** values;

Step2: Update the velocities and positions of particles according to formula **(1)** and formula **(2)**;

Step 3: Calculate the adaptive value of each particle;

Step 4: Compare the adaptive value of each individual particle with the adaptive value of the best position **pbest** experienced by it; if the new position is better than the original position, please update the original **pbest** ;

Step 5: Compare the adaptive value of each individual particle with the adaptive value of the best position **gbest** experienced by the particle swarm; it there is a better position, please update the original **gbest**;

Step 6: Check whether the termination condition (generally speaking, the algorithm has reached the maximum number of iterations, the optimal solution of particle swarm does not change after iteration for several times, or the optimal solution searched has reached the minimum adaptive threshold) is satisfied; terminate iteration of the algorithm if one condition is met; otherwise, return to Step 2 for further iteration.

### 2) Discrete particle swarm optimization algorithm

In the continuous space, the optimal position of individual particle and the optimal position of particle swarm will influence the particle velocity and guide the particles to approach the two points. In discrete combinational optimization problem, the state of particle is expressed with a sequence of integers, and the connotations of various symbols in the particle updating equation cannot maintain the connotations in continuous problem. A change is needed. Here the operation of particles is redefined by referring to the studies of Clerc[2] and other scholars.

**Definition 1** Commutator: Suppose that the particle state is $\mathbf{X(x_1, x_2, \cdots, x_n)}$; the operation of commutator $\mathbf{XO(x_i, x_j)}$ is defined as follows: to exchange the values corresponding to $\mathbf{x_i}$ and $\mathbf{x_j}$ positions in **X**.

For instance, $\mathbf{X = (1,2,3,4)}$ **and** $\mathbf{XO(x_i, x_j) = (0,1)}$; then $\mathbf{X' = X + XO(0,1) = (2,1,3,4)}$.

**Definition 2** Velocity: The list of one or multiple commutators is velocity,$\mathbf{V = (XO_1, XO_2, \cdots}$ **,XOn**.

**Definition 3** Addition Operation: The updating equation $\mathbf{X}_i' = \mathbf{X}_i + \mathbf{V}_i'$ of particles means to make commutators in $\mathbf{V}_i'$ act on position $\mathbf{X}_i$ in order and to get a new position $\mathbf{X}_i'$; addition of velocities $\mathbf{V}_i + \mathbf{V}_j$ means to connect the commutator list of the second velocity $\mathbf{V}_j$ to the end of the commutator list of the first velocity $\mathbf{V}_i$, and to treat the new list gained as the new velocity.

**Nature 1** Particle velocity has directivity.

For instance, $\mathbf{X} = (\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4})$, $\mathbf{V}_1 = (\mathbf{0}, \mathbf{1})(\mathbf{1}, \mathbf{2})$, and $\mathbf{V}_2 = (\mathbf{1}, \mathbf{2})(\mathbf{0}, \mathbf{1})$ ; then $\mathbf{X}' = \mathbf{X} + \mathbf{V}_1 = (\mathbf{2}, \mathbf{3}, \mathbf{1}, \mathbf{4})$ and $\mathbf{X}'' = \mathbf{X} + \mathbf{V}_2 = (\mathbf{3}, \mathbf{1}, \mathbf{2}, \mathbf{4})$ . Therefore, if the order of commutators is changed, totally different results might be gained.

**Definition 4** Subtraction Operation: The difference between positions of two particles is $\mathbf{X}' - \mathbf{X}$, and a velocity can be gained.

**Definition 5** Multiplication Operation: For the real number $\mathbf{c} \in (\mathbf{0}, \mathbf{1})$, length of the velocity $\mathbf{v}$ is supposed as $\mathbf{K}$ commutators; then $\mathbf{cV}$ means to cut the velocity list and to take the first $\mathbf{ck}$ (INT) commutators as the new velocity.

With the above definitions, calculation can be conducted for particles in discrete problems according to the updating formula $(\mathbf{1})$ and $(\mathbf{2})$ of standard PSO algorithm.

## III. IMPROVED ALGORITHM AEPSO-SA

### 1) Dynamically adjusted inertia weight

Shi and Eberhart studied the influence of inertia weight on the searching ability of PSO algorithm[3]; they discovered that when the value of $\boldsymbol{\omega}$ was large, global searching ability of the algorithm would be strong and there was a high possibility to jump out from the local optimal solution; when the value of $\boldsymbol{\omega}$ was small, the particle movement was mainly affected by individual optimal solution and group optimal solution, the local searching ability would be strong, and it was beneficial to algorithm convergence. The balance between local searching ability and global searching ability of particles in PSO algorithm can be controlled by setting the inertia weight. Shi and Eberhart proposed a linear decreasing method of dynamically adjusting the inertia weight according to the number of iterations. At the initial stage of algorithm operation, a large inertia weight will be used to guarantee global searching ability of the algorithm; later a small inertia weight will be used to enhance local searching ability of the algorithm and to accelerate convergence of the algorithm. Linear decreasing inertia weight PSO algorithm is one of the common PSO algorithms applied at present.

The strategy of linear decreasing inertia weight can improve the algorithm effect to some extent, but it still has some defects. Firstly, if the value of inertia weight $\boldsymbol{\omega}$ decreases too fast, the algorithm might fall into local searching before finding the position of extreme point. At the same time, if the value of $\boldsymbol{\omega}$ is high at the initial stage, it might directly skip the global optimal region when searching the global optimal region at the earlier stage. As a result, the algorithm accuracy will drop. Secondly, in each iteration process, all particles use the same inertia weight, which will make the difference among particles reduce in the iteration process. Thereby, the particle velocity will decline gradually and tend to be 0 ultimately. It can fall into local optimal solution easily. Therefore, the adjustment for the value of inertia weight ω should not only rely on the number of iterations but also depend on the evolution degree of particle population. Different values of inertia weight $\boldsymbol{\omega}$ should be used for particles of different evolution degrees, so as to increase the difference among particles and to better control the balance between global searching ability and local searching ability of particles. Thus a better searching effect can be realized. By aiming at this, this paper proposes an algorithm of dynamically adjusting inertia weight, i.e. AEPSO.

In the optimization process of particle swarm, each particle will approach the position of group optimal solution. By centering on the optimal solution $\mathbf{gbest}$ of particle swarm, the distance between each particle and the central point is calculated as the judgment basis for the evolution degrees of particles. Different inertia weights will be set for particles of different evolution degrees. The specific setting is as follows: In the iteration process, the distance $(\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N)$ between each particle and the central pointgbest is calculated; the distances are arranged in an ascending order and the particles are numbered according to the distances; number of the particle which is the closest to the central position is 0, and number of the particle which is the furthest to the central position is $N-1$; the array $\mathbf{locat}[\mathbf{N}]$ is used to record the number of each particle. The control factor $\mathbf{u_i}$ for inertia weight of each particle is calculated according to the number, and the specific calculation formula is as follows:

$$\mathbf{u_i} = \begin{cases} \boldsymbol{\alpha}, & \mathbf{locat[i]} < \mathbf{k_1 N} \\ \mathbf{1}, & \mathbf{k_1 N} \le \mathbf{locat[i]} \le \mathbf{k_2 N} \\ \boldsymbol{\beta}, & \mathbf{locat[i]} > \mathbf{k_2 N} \end{cases} \quad (3)$$

In the formula, $\mathbf{k}_1$ and $\mathbf{k}_2$ are control parameters used to divide the evolution degrees of particles, and $\mathbf{k_1} < \mathbf{k_2} < \mathbf{1}$; $\mathbf{N}$ means the scale of particles; $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are adjustment parameters which represent the inertia weight change ratio of particles of different evolution degrees. Inertia weight will be updated according to the following formula:

$$\boldsymbol{\omega_i} = \mathbf{u_i}\left(\boldsymbol{\omega_{max}} - (\boldsymbol{\omega_{max}} - \boldsymbol{\omega_{min}})\frac{\mathbf{gen}}{\mathbf{MAX\_GEN}}\right) \qquad (4)$$

In the formula, $\boldsymbol{\omega}_{\mathbf{max}}$ and $\boldsymbol{\omega_{min}}$ are initial value and final value of inertia weight; $\mathbf{gen}$ and $\mathbf{MAX\_GEN}$ are the current number of iterations and maximum number of iterations.

**2) Local searching strategy with simulated annealing idea**

When the inertia weight $\boldsymbol{\omega}$ of particles is dynamically adjusted by calculating the distance from particles to the global optimal solution $\mathbf{gbest}$, the solving accuracy of particles and convergence rate of the algorithm are effectively improved. However, the defects of PSO algorithm are not solved. The convergence rate of PSO algorithm is high at the initial stage and particles sway in a sine wave state. Under the influence of global optimal solution, particles approach the global optimal solution gradually. With the evolution of particles, when the positions of particles become close to the global optimal solution, the particle velocity will decrease and even turn into 0. In this way, particles might stay near the optimal solution rather than find global optimal solution of the problem. It can find a local optimal solution only. When the velocity of particle swarm becomes 0, particles will lose evolution ability. As a result, the algorithm cannot jump out from the local optimal solution and find the global optimal solution.

Simulated annealing algorithm conducts searching in the neighborhood of the solution, and this is a local searching strategy. However, simulated annealing algorithm might accept an inferior solution. Thus flexibility of the algorithm is increased and the hunting zone is expanded. The algorithm has a strong ability to enter the region of global optimal solution and to jump out from the local optimal solution. However, if the problem involves a large scale, it has to search the neighborhood of the current solution, which will reduce the global searching ability of simulated annealing algorithm. In order to reach the region of global optimal solution, a long time is needed.

Several second-best solutions are often distributed in the realm of an optimal solution, so strengthening the searching ability for regions near second-best solutions will provide a higher possibility to find the optimal solution. Based on this, this paper adds a local searching strategy into PSO algorithm. A threshold value $\mathbf{COOL}$ is set for the number of iterations by referring to the idea in simulated annealing algorithm. If the current global optimal solution does not change when the number of iterations exceeds $\mathbf{COOL}$, the algorithm will search the region of individual optimal solutions, and start an annealing process. In this way, places near the local optimal solutions can be searched, and the algorithm will jump out from these local optimal solutions at the same time.

Steps of AEPSO-SA algorithm proposed in this paper are as follows:

Step 1: Set acceleration constants of the algorithm and initialize a group of particles and velocities. The optimal position $\mathbf{pbest}$ experienced by each individual particle is set as the initial position, and the optimal position $\mathbf{gbest}$ experienced by the particle swarm is the optimal value among all $\mathbf{pbest}$ values. Set the initial temperature as $\mathbf{t_0}$. The iterative step of each temperature is $\mathbf{DN}$ and the threshold value of annealing process is $\mathbf{COOL} = \mathbf{0}$;

Step 2: Calculate the distance between each individual particle and the global optimal position, arrange the distances in order, number the particles according to the order, and calculate the inertia weights of particles according to formula (**3**) and (**4**);

Step 3: Update the velocities and positions of particles according to formula (**1**) and formula (**2**);

Step 4: Calculate the adaptive value of each particle;

Step 5: Compare the adaptive value of each particle with the adaptive value of the best position **pbest** experienced by it; if the new position is better than the original position, please update the original **pbest**;

Step 6: Compare the adaptive value of each individual particle with the adaptive value of the best position **gbest** experienced by the particle swarm; it there is a better position, please update the original **gbest** and meanwhile set the value of **COOL** as 0;

Step 7: Inquire the value of **COOL** and judge whether simulated annealing is needed. If the value of **COOL** reaches the set value, enter Step 8; otherwise, go to Step 9;

Step 8: Start a simulated annealing process and conduct neighborhood searching for the optimal positions **pbest** of all individual particles for **DN** times. If a solution better than the current global optimal position **gbest** is found, update the global optimal position **gbest**. As for new solutions searched, judge whether to replace the individual optimal position **pbest** according to Metropolis criterion. Set the value of **COOL** as 0 after simulated annealing process;

Step 9: Update the temperature according to the number of iterations, and add 1 to the value of **COOL**;

Step 10: Check whether the termination condition (generally speaking, the algorithm has reached the maximum number of iterations, the optimal solution of particle swarm does not change after iteration for several times, or the optimal solution searched has reached the minimum adaptive threshold) is satisfied; terminate iteration of the algorithm if one condition is met; otherwise, return to Step 2 for further iteration.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

### 1) Experimental results

In order to verify the effectiveness of AEPSO-SA algorithm, AEPSO-SA algorithm is tested by utilizing the travelling salesman problem. Travelling salesman problem (TSP) is an issue to seek the optimal path. This is an issue to solve a path to different cities that can minimize the total travelling cost when a traveller wants to travel to many cities by starting from one city and the traveller will return to the first city after visiting all cities. City problem TSP14 (Burma14), city problem 30 (Oliver30) and city problem 48 (att48) are adopted for test in the experiment. Experimental results of the following algorithms are compared with the results of AEPSO-SA algorithm: linear decreasing inertia weight PSO algorithm, adaptive inertia weight AEPSO algorithm mentioned in this paper, and simulated annealing (SA)algorithm. Parameter setting of various algorithms is as follows:

Linear decreasing inertia weight PSO algorithm: The maximum number of iterations $\mathbf{MAX_{GEN} = 2000}$; particle scale $\mathbf{N = 50}$; $\boldsymbol{\omega}_{\mathbf{max}} = \mathbf{0.95}$; $\boldsymbol{\omega}_{\mathbf{min}} = \mathbf{0.4}$; $\mathbf{C_1 = C_2 = 1}$.

SA algorithm: The initial temperature$\mathrm{TT} = 250.0$; cooling times: $\mathbf{T = 400}$; coefficient of temperature drop $\mathbf{aa = 0.98}$; iterative step under each temperature:$\mathbf{N = 40}$.

AEPSO algorithm: $\mathbf{MAX_{GEN} = 2000}$; $\mathbf{k_1 = \frac{1}{7}}$; $\mathbf{k_2 = \frac{4}{7}}$; $\boldsymbol{\alpha} = \mathbf{1.5}$; $\boldsymbol{\beta} = \mathbf{0.8}$.

AEPSO-SA algorithm: $\mathbf{MAX_{GEN} = 1000}$; other parameters are the same with the above two algorithms.

Each algorithm is tested for 30 times, and the test results are shown in Table 1.

Table 1 Test results of various algorithms

| Problem | Existing optimal solution | Algorithm | Average solution | Optimal solution | Worst solution | Times of acquiring the optimal solution |
|---|---|---|---|---|---|---|
| Burma14 | 30.8785 | PSO | 31.32403 | 30.8785 | 32.54418 | 8 |
| | | AEPSO | 31.07592 | 30.8785 | 32.15857 | 18 |
| | | SA | 30.8785 | 30.8785 | 30.8785 | 30 |
| | | AEPSO-SA | 30.8785 | 30.8785 | 30.8785 | 30 |
| Oliver30 | 423.74 | PSO | 609.358 | 527.378 | 683.494 | 0 |
| | | AEPSO | 577.346 | 485.935 | 669.287 | 0 |
| | | SA | 428.654 | 423.74 | 441.103 | 7 |
| | | AEPSO-SA | 424.105 | 423.74 | 425.510 | 22 |
| att48 | 10628 | PSO | 21508.7 | 18931 | 25971 | 0 |
| | | AEPSO | 18827.9 | 16079 | 22092 | 0 |
| | | SA | 11042.4 | 10752 | 11366 | 0 |
| | | AEPSO-SA | 10686.9 | 10628 | 10782 | 3 |

### 2) Performance analysis

According to data in Table 1, for city problem 14 with a small scale, all the 4 algorithms can gain a good result; SA algorithm and AEPSO-SA algorithm can obtain the global optimal solution every time. When the city scale expands to 30, standard PSO algorithm falls into the local optimal solution easily; the solving accuracy of improved AEPSO algorithm is increased, but the result is not ideal. SA algorithm and AEPSO-SA algorithm can gain the global optimal solution for multiple times, but the results of SA algorithm fluctuate greatly and the times of acquiring the global optimal solution are much fewer when compared with AEPSO-SA algorithm. When the problem scale expands to 48, both PSO algorithm and AEPSO algorithm can hardly gain a good solution; SA algorithm and AEPSO-SA algorithm can get a value close to the global optimal solution. SA algorithm does not gain the global optimal solution, but AEPSO-SA algorithm obtains the global optimal solution for three times. Moreover, AEPSO-SA algorithm is better than SA algorithm in the average solution and worst solution.

## V. CONCLUSIONS

This paper analyzes the key role of inertia weight in PSO algorithm, and introduces a strategy of dynamically adjusting the inertia weight. Meanwhile, by aiming at the weakness of PSO algorithm in local searching ability, simulated annealing idea is added into the algorithm as a local searching strategy. An improved hybrid PSO algorithm, AEPSO-SA is proposed. According to the experimental results, the hybrid algorithm possesses higher solving accuracy and stronger ability of jumping out from the local optimal solution than PSO algorithm and simulated annealing algorithm. Therefore, it is an improved algorithm with relatively high efficiency.

## REFERENCES

[1] Kennedy J. Particle swarm optimization[J]. Proceedings of IEEE International Conference on Neural Networks, 1995, 4(8):129-132.

[2] Clerc M, Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1):58-73.

[3] Shi Y, Eberhart R. A modified particle swarm optimizer[C]//Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998: 69-73.

[4] REN Zihui and WANG Jian. New Adaptive Particle Swarm Optimization Algorithm with Dynamically Changing Inertia Weight[J].Computer Science, 2009, 36(2): 227-229.

[5] LI Zhiyong, MA Liang, and ZHANG Huizhen.Adaptive Cellular Particle Swarm Algorithm for Solving 0/1 Knapsack Problem[J].Computer Engineering, 2014, 40 (10): 198-203.

[6] WANG Fang. Hybrid Algorithmof Particle Swarm and Simulated Annealing and its Application in Logistics Distribution[D].East China University of Science and Technology, 2011.

[7] WANG Yonggui, LIN Lin, and LIU Xianguo. Research on Text Clustering Algorithm Based on Improved Particle Swarm Optimization[J].Computer Engineering, 2014, 40(11): 172-177.