EasyChair Preprint

№ 11312

Task Scheduling with Improved Particle Swarm Optimization in Cloud Data Center

Yang Bi, Wenlong Ni, Yao Liu, Lingyue Lai and Xinyu Zhou

November 17, 2023

Task Scheduling with Improved Particle Swarm Optimization in Cloud Data Center

Yang Bi[1], Wenlong Ni[1], Yao Liu[1], Lingyue Lai[1], and Xinyu Zhou[1]

School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, CHINA
{byang, wni, liuy, laily, xyzhou}@jxnu.edu.cn

Abstract. This paper proposes an improved particle swarm optimization algorithm with simulated annealing (IPSO-SA) for the task scheduling problem of cloud data center. The algorithm uses Tent chaotic mapping to make the initial population more evenly distributed. Secondly, nonlinear adaptive inertia weights is incoporated to adjust optimization seeking capabilities of particles in different iteration periods. Finally, the Metropolis criterion in SA is used to generate perturbed particles, combined with an modified equation for updating particles to avoid premature particle convergence. Comparative experimental results show that the IPSO-SA algorithm improves 13.8% in convergence accuracy over the standard PSO algorithm. The respective improvements over the other two modified PSO are 15.2% and 9.1%.

Keywords: Cloud Data Center · Task Scheduling · Particle Swarm Optimization · Simulated Annealing.

1 Introduction

In cloud data centers, improving computing efficiency and reducing resource costs is always a challenging problem. Due to the heterogeneity and dynamics of cloud environments, resource allocation and task scheduling is considered as a NP hard problem, which is a non-deterministic problem of polynomial complexity. For such problems with long solution time and high complexity, it is intuitive to use simulated annealing algorithm, genetic algorithm, ant colony algorithm, particle swarm algorithm, etc. However, such algorithms may have the following disadvantages like many parameters to adjust, high computational cost and hard to implement, etc.

PSO originated from the research on the foraging behavior of birds, and was first proposed by Dr. Eberhart and Dr. Kennedy [1]. The original purpose of this algorithm is to achieve the study of complex social behavior by simulating a simple social system [2]. After further research, it was found that the particle swarm algorithm can be used to solve complex optimization problems. A particle in the swarm is a candidate solution. Through the cooperation and information sharing among individuals in the swarm, the optimal solution to the problem to be optimized is found in the search space of a given dimension. PSO only has

three control parameters: inertia weight, cognitive acceleration coefficient, and social acceleration coefficient. A small change in any of these three parameters will bring about a difference in algorithm performance as shown in[3] and [4].

Although the PSO algorithm has good optimization performance, it still has serious premature convergence problems, which leads to defects such as low convergence accuracy. There are four main strategies to improve the standard particle swarm algorithm: modifying the control parameters of the particle swarm algorithm, mixing the particle swarm algorithm with genetic algorithm , differential evolution algorithm and other meta-heuristic algorithms, collaboration and multi-swarm technology [5]. For the inertia weights in the particle swarm algorithm, literature [6] proposed a method to represent inertia weights with random values. Because in some applications, it is not easy to predict the required The size of the inertia weight, so this method is suitable for solving the dynamic environment. [7] proposed a linearly changing inertial weight, which can endow particles with the ability of individual optimization and collective optimization in different iteration periods. For the acceleration coefficients in the PSO, [8] proposed a hierarchical particle swarm algorithm with time-varying acceleration coefficients, using the number of iterations Dynamically adjust the particle swarm acceleration coefficient to improve the development and search capabilities of particles in the population. Literature [9] uses a mixture of PSO and genetic algorithms to solve multimodal problems. Literature [10] divides the population into a master population and multiple subordinate populations, constituting a multi-group cooperative particle swarm algorithm.

Based on the particle swarm algorithm and simulated annealing algorithm, IPSO-SA is proposed in this paper to be used in cloud computing scenarios, with a multi-objective optimization considering both cloud computing execution time and resource cost. It is implemented from the following aspects:

1. The Tent chaotic map is used to initialize the particle swarm population to improve the traversal of the population.
2. The inertia weights are adjusted nonliearly which can strengthen the optimization ability of the particles in different iteration periods.
3. Hybrid simulated annealing algorithm is adopted, using the Metropolis criterion to disturb the population, thus improving the particle swarm velocity update equation to avoid particles falling into local optimal solutions.

2 Problem Description

The cloud computing task scheduling model can be abstracted as transferring a large number of tasks to the data center and allocating them to different VMs in the resource pool through task execution resource allocation strategies. An appropriate allocation strategy can not only reduce the cost of cloud service providers, but also improve user satisfaction by reducing waiting time. A mathematical model for multi-objective optimization is established for task scheduling.

2.1 System Model

A virtual machine (VM) must be deployed on a PM in the Data Center. In this paper, we suppose that the number of VMs is n, $VM = \{vm_1, vm_2, vm_3, \ldots, vm_n\}$. The deployment and operation of each VM node is independent of each other. The VM nodes also have resources such as CPU, memory, storage, GPU and bandwidth. In this paper, we are mainly concerned with the computing and bandwidth resources of VMs. For example, $vm_j =< VMe_j, VMt_j >$ $(j = 1, 2, 3, \ldots, n)$, where VMe and VMt are the computing power and bandwidth resources.

Each task is assigned to a VM for processing. Task scheduling sends each task to a selected VM for best performance. Assuming the total number of pending tasks is k, so the set of tasks is $T = \{t_1, t_2, t_3, \ldots, t_m\}$. In addition, $t_i =< Tl_i, Td_i >$ $(i = 1, 2, 3, \ldots, m)$. Where Tl is the task length of t_i, which is proportional to the computation time of tasks. Td is the amount of data to be transferred for t_i, which is proportional to the transfer time of the task. Let x_{ij} be a binary variable indicating whether t_i is assigned on vm_j, the definition rules are as follows:

$$x_{ij} = \begin{cases} 1\ , t_i \text{ assign to } vm_j, \\ 0\ , \text{other}. \end{cases}$$

In the above equation, the constraint on x_{ij} is:

$$\sum_{j=1}^{n} x_{ij} = 1.$$

2.2 Fitness Function

The optimization objective of this paper is to minimize task execution time and execution cost. Therefore the total computation time, the total transmission time, the maximum end time and the execution cost of the task need to be defined. The details are as follows.

1. *Total computation time of tasks.* According to the allocation strategy, t_i is assigned to the vm_j to perform computing tasks. Therefore, the computation time of the task is defined as follows:

$$ET_i = \sum_{j=1}^{n} x_{ij} \times \frac{Tl_i}{VMe_j}. \tag{1}$$

The total calculation time of a task is the sum of the time required to complete the task calculation by assigning a single task to a VM, so it is defined as follows:

$$ET = \sum_{i=1}^{m} ET_i. \tag{2}$$

Where, n is the total number of VMs and m is the total amount of tasks. The following is similar and will not be repeatedly defined.

2. *Total transfer time of tasks.* Similar to the execution time, the transmission time of task data is related to the bandwidth resource of the VM. Therefore, the transmission time of the t_i is defined as follows:

$$TT_i = \sum_{j=1}^{n} x_{ij} \times \frac{Td_i}{VMt_j}. \tag{3}$$

The total task transfer time is the sum of all the individual task transfer times in the task sequence and is therefore defined as follows:

$$TT = \sum_{i=1}^{m} TT_i. \tag{4}$$

3. *The maximum ending time of tasks.* The time required for a complete cloud service is equal to the release time of the last VM in the cloud service, and the release time of a VM is:

$$T_j = \sum_{i=1}^{m} x_{ij} \times \left(\frac{TL_i}{VMe_j} + \frac{Td_i}{VMt_j} \right), (j = 1, 2, 3 \ldots, n). \tag{5}$$

Therefore, the maximum end time of the cloud service is:

$$T = \max\left(T_1, T_2, T_3, \ldots, T_n\right). \tag{6}$$

4. *Resource Occupation Cost.* Since the task set is accepted by the virtual node, the resource occupation cost is generated, so the following equation is defined:

$$C = (ET + TT) \times q. \tag{7}$$

The equation, q is the cost coefficient of cloud service per unit time.

Based on the above optimization objectives and problem description, the adaptation function constructed in this paper is as follows.

$$F = \alpha_1 f_1(x) + \alpha_2 f_2(x), \tag{8}$$

where $f_1(x)$ is the minimized task execution time, and $f_2(x)$ is the minimized resource cost, described as follows:

$$f_1(x) = \min(T), f_2(x) = \min(C). \tag{9}$$

Furthermore, in equation (8), α_1 and α_2 are the weight coefficients of the objective function. The difference between the weight coefficients determines the optimization focus of the resource allocation strategies.

3 Proposed IPSO-SA Algorithm

Based on PSO, this paper proposes IPSO-SA. This section is mainly concerned with the encoding and decoding of particles, and three major improvement strategies. These are Tent mapping initial population, non-linear dynamic adaptive inertia weights, and hybrid simulated annealing algorithm to improve the particle swarm update formulation, respectively.

3.1 Encoder and Decoder

Coding is used to map the position of particles to the solution space. In this paper, continuous position information is realized through the principle of minimum position (SPV) mentioned in [11], which assumes that the position information of the generation i particle is:

$$X_i = (x_1, x_2, \ldots, x_n).$$

The particles encoded using SPV is:

$$\lfloor |X_i| \rfloor = (\lfloor |x_1| \rfloor, \lfloor |x_2| \rfloor, \ldots, \lfloor |x_n| \rfloor).$$

For example, suppose there are 10 tasks are assigned to 5 VMs for processing. If the position information of the particle d in the iteration i is $\{2.34, 0.56, -4.3, 1.2, -1.56, 3.16, 4.56, -0.13, 2.09, -1.3\}$, the encoded position information is $\{2, 0, 4, 1, 1, 3, 4, 0, 2, 1\}$, and the corresponding allocation strategy is shown in the Table 1.

Table 1. Task scheduling strategy.

TaskID	1	2	3	4	5	6	7	8	9	10
VMID	2	0	2	1	1	3	4	0	2	1

3.2 Algorithm Improvements

The standard PSO algorithm defines the equation for updating the velocity and position of population particles [1]. Particle i updates its velocity and position according to equation (10) and (11) in the $t+1$ iteration.

$$v_{in}^{t+1} = \omega v_{in}^{t} + c_1 \times r_1 \times \left(p_{in} - x_{in}^{t}\right) + c_2 \times r_2 \times \left(g_{in} - x_{in}^{t}\right), \tag{10}$$

$$x_{in}^{t+1} = x_{in}^{t} + v_{in}^{t+1}. \tag{11}$$

Where c_1 and c_2 are the cognitive acceleration coefficient and social acceleration coefficient, and r_1 and r_2 are two uniform random values generated within $[0, 1]$ interval. p_{in} is the individual optimal value of the current particle, and g_{in} is the overall optimal value of the population where the particle is located.

In the cloud scenario, PSO algorithm has the defects of unreasonable population initialization, easy convergence and premature, low convergence accuracy, and easy to fall into individual optimal solution [12]. This paper improves from the following aspects.

1. *Tent Map Initialization Population.* In the initial stage of PSO, the usual way is to use random functions to generate the initial information of particles. However, in the cloud scenario, it is necessary to use each VM as evenly as possible. At this time, the above method becomes less applicable. Therefore, in this paper, tent map is used to initialize the position and velocity of particles, as described in the following equation:

$$x_{n+1} = \begin{cases} \frac{x_n}{\alpha} & , (0 \le x_n < \alpha) , \\ \frac{1-x_n}{1-\alpha} & , (\alpha < x_n \le 1) . \end{cases}$$

Figure 1 shows the chaos values generated by the Tent Map. As can be seen from the figure, the sequence generated by the Tent Map is well distributed and random.

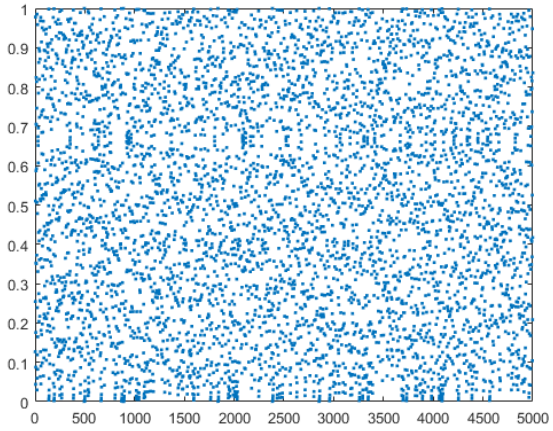

Fig. 1. Tent Chaos Map Chaos Value Distribution.

2. *Inertial Weights.* The ω in the PSO algorithm is called the inertia weight and is usually set to a fixed value. But, the results of the literature [12] show that when the number of iterations increases, taking a fixed-value solution leads to an amplification of many of the details of the problem that need to be solved. In addition, [12] indicates that, when $0 < \omega < 1$, particles gradually converge with the increase of iteration times. Larger ω are beneficial for finding the overall optimal solution. Smaller ω are beneficial for finding individual optimal solutions.
According to the fitness function in this paper, we can know that the optimization problem in this paper belongs to the minimum value problem, so ω should gradually decrease with the increase of the number of iterations. Based on the above theory, this article adopts a nonlinear decreasing function to optimize ω. The value of ω can be changed adaptively during the iteration process. The improved nonlinear adaptive inertia weight iteration

equation is as follows.

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \times e^{\left(-\frac{2\times t}{T_{\max}}\right)}. \tag{12}$$

In the equation(12), $\omega_{\max}$ is the maximum inertia weight. $\omega_{\min}$ is the minimum inertia weight. t is the current iteration number, and $T_{\max}$ is the maximum iteration number.

3. *Metropolis Criterion.* The Metropolis criterion is one of the core factor of SA algorithms [13], which does not use completely deterministic rules but chooses to accept new states with probability. In the t-th iteration, the algorithm will randomly generate a new solution g'_{in} in the neighborhood of all optimal g_{in} obtained by the particle swarm algorithm, and then determine whether to accept the solution through the Metropolis criterion. The Metropolis criteria are as follows:

$$\Delta F = F\left(g'_{in}\right) - F\left(g_{in}\right), \tag{13}$$

$$p\left(g_{in} \to g'_{in}\right) = \begin{cases} 1 & , \Delta F < 0, \\ e^{-\frac{\Delta F}{T}} & , \Delta F > 0. \end{cases} \tag{14}$$

Where ΔF is the difference value of fitness function, T is the current temperature, which is changed according to the following equation.

$$\begin{aligned} T_0 &= -\frac{F\left(g_{0n}\right)}{\log(0.2)}, \\ T &= K \times T_0. \end{aligned} \tag{15}$$

Where T_0 is the initial temperature, K is the cooling coefficient. At this point, equation(10) is improved to (15). Using the Metropolis criterion, we improve equation 11 and 11 as follows.

$$v_{in}^{t+1} = \omega \times v_{in}^{t} + c_1 \times r_1 \times \left(p_{in} - x_{in}^{t}\right) + c_2 \times r_2 \times \left(g_{in} - x_{in}^{t}\right) + c_3 \times r_3 \times \left(g'_{in} - x_{in}^{t}\right). \tag{16}$$

Where, c_1, c_2 and c_3 are the particle acceleration coefficients. The constraint conditions for using new solutions generated within the neighborhood as perturbed particles are:

$$p\left(g_{in} \to g'_{in}\right) > \text{rand}(0, 1).$$

In addition, a contraction factor is introduced to ensure the contractility of the population particles, and the particle position is updated according to equation(17).

$$\begin{aligned} \chi &= \frac{2}{\left|2 - C - \sqrt{C^2 - 4 \times C}\right|}, \\ x_{in}^{t+1} &= x_{in}^{t} + \chi v_{in}^{t+1}. \end{aligned} \tag{17}$$

4 Simulation Results

In this paper the CloudSim software is adopted for simulation experiments, which is developed by [14]. Table 2 shows the experimental parameter configuration.

Table 2. VMs configuration table

Parameter	Value
Processing speed of VMs/ MIPS	[200,600]
Bandwidth of VMs/ Mbps	[1000,2500]
Memory of VMs/ GB	1.70
Number of VMs	20
Number of tasks	200
Task length	[25000,250000]
Task data volume	[100,600]

In addition, the data for the VMs and tasks to be used in this paper were randomly generated based on the Table 2.

This paper uses the standard PSO, IPSO_1 in [8] and IPSO_2 [15], for comparison experiments with the proposed IPSO-SA. The strategy in [8] uses an adaptive acceleration factor, and [15] uses an inertial weight curve descent strategy. The population size is set to 25 and the number of iterations is 1000 for all the above four algorithms. The values of other parameters are referred to Table 3.

Table 3. Algorithm parameter

PSO	inertia weight	0.9
	learning factor	$c_1 = c_2 = 2.05$
IPSO_1	inertia weight	0.9
	learning factor	$c_{\max} = 2.5, c_{min} = 0.5$
IPSO_2	inertia weight	$\omega_{\max} = 0.9, \omega_{\min} = 0.2$
	learning factor	$c_1 = c_2 = 2.05$
	Inertia Curve Parameters	-0.95
IPSO-SA	inertia weight	$\omega_{\max} = 0.9, \omega_{\min} = 0.2$
	learning factor	$c_1 = c_2 = 2.05, c_3 = 0.5$
	The initial temperature	$T = 1000000$
	cooling coefficient	$K = 0.998$

The comparison experiments are performed several times under the above parameter conditions. The algorithm performance comparison is shown in Figure 2.

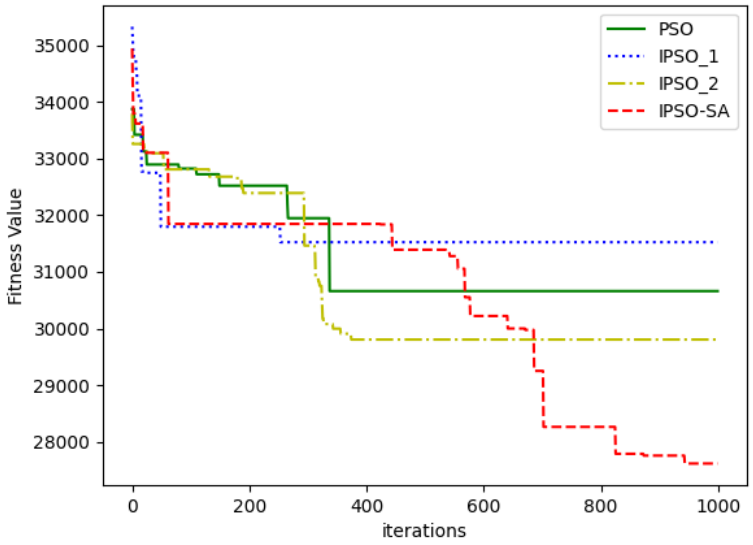

Fig. 2. Comparison of algorithm performance.

Fifty comparison experiments were conducted and the convergence accuracy of each algorithm is shown in the Table 4.

Table 4. Comparison of algorithm convergence accuracy

Algorithm	Maximum Fitness	Minimum Fitness	Average Fitness
PSO	31969.61723	29618.40511	31128.05389
IPSO_1	32276.67126	31023.48934	31635.17231
IPSO_2	31776.67126	27705.92521	29512.43004
IPSO-SA	29749.45768	23275.83437	26819.8504

According to Equation 8, the optimisation objective of this paper lies in minimising the execution time and execution resources of the task. Therefore, the smaller the fitness value, the better the scheduling strategy.

Based on the results of the above-mentioned multiple comparison experiments, the IPSO-SA algorithm, after introducing simulated annealing perturbed particles, better avoids the defects of premature convergence of the PSO. In terms of algorithm convergence accuracy, IPSO-SA improves 13.8% over PSO. It is 15.2% better than IPSO_1. The improvement over IPSO_2 is 9.1%. Figure 3 shows a comparison of the average fitness values of the four algorithms.

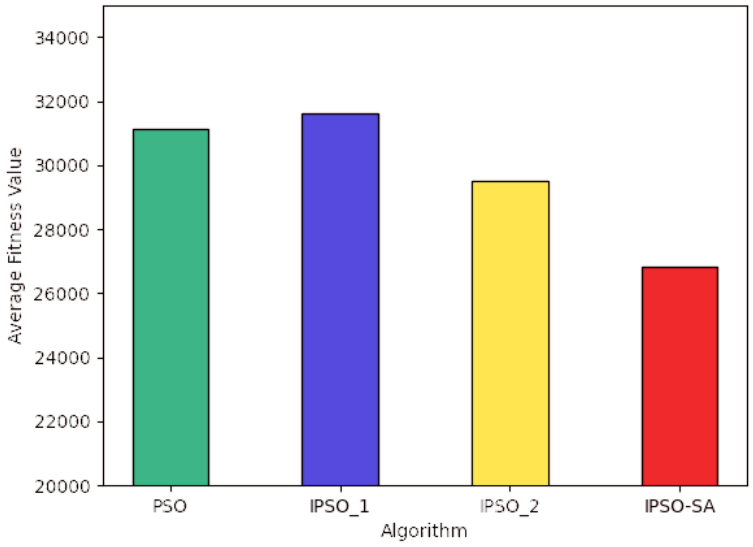

Fig. 3. Comparison of average fitness values.

Combining the analysis results in Figure 2 and Table 4, it can be concluded that the IPSO-SA algorithm have better convergence accuracy and at the same time, and reducing the probability of premature convergence.

5 Conclusion

In this paper, we propose the improved PSO algorithm, which is used to solve the task scheduling problem in cloud data centers. Firstly, the initial population is made ergodic by Tent chaotic mapping. Then, construct non-linear adaptive functions that dynamically adjust the particle's ability to find the best at different times. Finally, the Metropolis criterion in SA was introduced to generate perturbed particles, using an improved velocity and position update formulation to reduce the probability of premature convergence of the algorithm. Comparative experimental results show that the IPSO-SA proposed in this paper has high convergence accuracy, and the particles do not remain in a locally optimal solution for long.

In future work, we will consider improving the convergence speed of the algorithm as a new optimization goal. Meanwhile, we consider applying the swarm intelligence algorithm to more application scenarios.

References

1. Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Proceedings of ICNN'95-international conference on neural networks. Vol. 4. IEEE, 1995.
2. Garnier, Simon, Jacques Gautrais, and Guy Theraulaz. "The biological principles of swarm intelligence." Swarm intelligence 1 (2007): 3-31.
3. Eltamaly, Ali M. "A novel strategy for optimal PSO control parameters determination for PV energy systems." Sustainability 13.2 (2021): 1008.
4. Harrison, Kyle Robert, Andries P. Engelbrecht, and Beatrice M. Ombuki-Berman. "Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm." Swarm and evolutionary computation 41 (2018): 20-35.
5. Shami, Tareq M., et al. "Particle swarm optimization: A comprehensive survey." IEEE Access 10 (2022): 10031-10061.
6. Li, Mi, et al. "A multi-information fusion "triple variables with iteration" inertia weight PSO algorithm and its application." Applied Soft Computing 84 (2019): 105677.
7. Shi, Yuhui, and Russell C. Eberhart. "Empirical study of particle swarm optimization." Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). Vol. 3. IEEE, 1999.
8. Ratnaweera, Asanga, Saman K. Halgamuge, and Harry C. Watson. "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients." IEEE Transactions on evolutionary computation 8.3 (2004): 240-255.
9. Kao, Yi-Tung, and Erwie Zahara. "A hybrid genetic algorithm and particle swarm optimization for multimodal functions." Applied soft computing 8.2 (2008): 849-857.
10. Niu, Ben, et al. "MCPSO: A multi-swarm cooperative particle swarm optimizer." Applied Mathematics and computation 185.2 (2007): 1050-1062.
11. Alguliyev, Rasim M., Yadigar N. Imamverdiyev, and Fargana J. Abdullayeva. "PSO-based load balancing method in cloud computing." Automatic Control and Computer Sciences 53 (2019): 45-55.
12. Parsopoulos, K. E., et al. "Improving particle swarm optimizer by function "stretching"", Nonconvex Optimization and Applications, vol. 54, ch. 3." (2001): 445-457.
13. Van Laarhoven, Peter JM, et al. Simulated annealing. Springer Netherlands, 1987.
14. Calheiros, Rodrigo N., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41.1 (2011): 23-50.
15. Lei, Kaiyou, Yuhui Qiu, and Yi He. "A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization." 2006 1st international symposium on systems and control in aerospace and astronautics. IEEE, 2006.