MULTI-OBJECTIVE OPTIMIZATION ALGORITHM BASED ON IMPROVED PARTICLE SWARM IN CLOUD COMPUTING ENVIRONMENT

MIN ZHANG*

College of Applied Science, Jiangxi University of Science and Technology, Ganzhou, China

GANG LI

School of Road and Bridge Engineering, Xinjiang Vocational & Technical College of Communications, Urumchi, China

ABSTRACT. In cloud computing environment, in order to optimize the deployment scheduling of resources, it is necessary to improve the accuracy of the optimal solution, guarantee the convergence ability of the algorithm, and improve the performance of cloud computing. In this paper, a multi-objective optimization algorithm based on improved particle swarm is proposed. A multi-objective optimization model is built. Improved multi-scale particle swarm is used to optimize the built multi-objective model. The combination of the global search capability and the local search capability of the algorithm is realized by using Gaussian variation operator with varied scales. The large scale Gaussian variation operator with concussion characteristics can complete fast global search for decision space, so that particles can quickly locate the surrounding area of the optimal solution, which enhances the ability to escape the local optimal solution of the algorithm and avoids the occurrence of premature convergence. The small scale variation operator gradually reduces the area near the optimal solution. Experimental results show that the improved particle swarm optimization algorithm can effectively improve the precision of the optimal solution and ensure the convergence of the algorithm.

1. Introduction. The world is entering the era of cloud computing. Due to the rapid development of computer technology and the rapid growth of data on the Internet, it is a great challenge to process and store these massive data [21]. Google has proposed the idea of cloud computing. Cloud computing provides powerful computing and massive data storage capacity to users through large-scale data centers, which is more economical and energy saving than general data center [6,22]. When an optimization problem contains only one objective equation to be optimized, such a multi-objective optimization problem is called as single objective optimization problem [4]. However, there often exists the problem of making multiple objectives satisfy the decision-maker in the specific region. Improving one goal may cause another goal to deteriorate [5,17]. Therefore, it has to make balance between multiple objectives. For a decision-maker, what is found is not a single solution, but a set of solutions that cannot be compared with each other. This solution set of is called

2010 Mathematics Subject Classification. 31C20.

Key words and phrases. Cloud computing environment, improved particle swarm, multi-objective, optimization algorithm.

* Corresponding author: Min Zhang.
as Pareto optimal solution set or non-dominated solution set. The existence of Pareto optimal solution set represents the most fundamental difference between the multi-objective optimization problem and the single objective optimization problem. It can be understood that Pareto optimal solution means that there will be no such solution: there is a solution with at least one objective optimal than the other solutions, while other objectives not worse than other solutions [3, 20]. For all objectives, the advantages and disadvantages of the solution members in Pareto optimal solution set cannot be compared. The reason is that there will not be a situation where some objectives are optimized and other objectives are not deteriorating. Many practical problems can be modeled as multi-objective optimization problems, and some unresolved problems in the field of multi-objective optimization still remain to be solved. Therefore, the research on the related problems of multi-objective optimization has become a very meaningful subject [9,15].

In the current discrete particle swarm optimization algorithm, the total energy consumption of the network and the load balancing of mobile agents is taken as fitness function to find approximate optimal solutions in the cooperative path planning of multiple mobile agents. Simulation results show that the proposed multiple mobile agents cooperative path planning performs better than other similar algorithms in terms of total network energy consumption and network lifetime, but this method has poor convergence ability.

2. Multi-objective optimization algorithm based on improved particle swarm.

2.1. Multi-objective optimization model. In order to create a clearer mathematical model for data distribution and task scheduling, some of the concepts are illustrated as follows. In processing interaction diagram, for the task \( T_i : i = \{1,2,\ldots,n\} \), there are \( n \) tasks. The amount of data of each task is mapped to a million instructions to measure the complexity of a task. \( \text{DP}_i \) is the amount of data to be processed for the task \( t_i \), \( DC_k : k = \{1,2,\ldots,n\} \) is the \( M \) data centers. The performance of each data center is measured by how many million processed instructions per second [7]. \( B_{kl} : kl = \{1,2,\ldots,m\} \) is the bandwidth between two data centers, \( m \) is the total number of data centers. \( DT_{ij} \) is the amount of the exchanged data between the task \( t_i \) and \( t_j \). The data is generated and required by the task \( t_i \), which is transmitted from the data center for the task \( t_i \) to the data center for the task \( t_j \).

Processing time is defined as the amount of data of a task divided by the processing capacity of the data center, given by Eq. (1). \( T_p \) is defined as the entire processing time of all data allocated to the corresponding data center [13,23]. In \( \sum_{i=1}^{n} \sum_{k=1}^{m} x_{ik}, n \) denotes the number of the tasks, \( m \) denotes the number of the data centers. If the task \( t_i \) is allocated to the data center \( k, x_{ik} = 1 \), otherwise, \( x_{ik} = 0 \). \( \sum_{i=1}^{n} \sum_{k=1}^{m} x_{ik} \) denotes \( n \) tasks allocated to \( m \) data centers. \( \frac{\text{DP}_i}{DC_k} \) denotes the processing time of the task \( t_i \) allocated to the data center \( DC_k \).

\[
T_p = \sum_{i=1}^{n} \sum_{k=1}^{m} x_{ik} \times \frac{\text{DP}_i}{DC_k} \tag{1}
\]

Transmission time is defined as the amount of data needed to be transmitted divided by the bandwidth between the data centers, given by Eq. (2). \( T_t \) represents the transmission time of data need to be transmitted from other data centers, when different tasks are allocated to the corresponding data centers [1].
\[ T_i = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} x_{ik} \times x_{jl} \times \frac{DT_{ij}}{B_{kl}} \] (2)

\[ T = T_p + T_t \] (3)

Amazon provides three types of charging mode. In this paper, the charging method on demand is used and computing power is purchased by hour. The cost of transmission is for data transmission between different data centers. The operating system is Linux/UNIX. The equation for calculating the cost is built according to the charging standard of data transmission to the data center. In Eq. (4), the processing cost is \( C_p \), which is the product of processing time \( T_p \) and processing cost \( P_k \) for unit data. \( P_{out} \) is defined as the charging standard of data transmission from the data center \( DC_k \) and \( P_{in} \) is the charging standard of data transmission to the data center. In Eq. (5), \( C_t \) denotes the transmission cost. In Eq. (6), \( C \) is the total cost.

\[ C_p = T_p + P_k \] (4)

\[ C_t = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} x_{ik} \times x_{jl} \times (DT_{ij} \times P_{out_k} + DT_{ij} \times P_{in}) \] (5)

\[ C = C_p + C_t \] (6)

Eq. (7) \sim Eq. (9) is constrained conditions. Eq. (7) represents a task must be allocated to a data center. Eq. (8) represents if two tasks are allocated to different data center and there is a requirement for data communication between them, \( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} x_{ik} \times x_{jl} = 1 \), otherwise, \( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l \neq k} x_{ik} \times x_{jl} = 0 \). Eq. (9) represents the value range of \( x_{ik} \) and \( y_{il} \) is 1 or 0. The value 1 represents a task is allocated to a data center, otherwise, 0.

\[ \sum_{k=1}^{m} x_{ik} = 1, \forall i = 1, 2, \ldots, n \] (7)

\[ \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l \neq k} x_{ik} \times x_{jl} = 1 \] (8)

\[ x_{ik}, y_{il} \in \{0, 1\}, \forall i, j = 1, 2, \ldots, n, \forall k, l = 1, 2, \ldots, m \] (9)

2.2. Multi-objective particle swarm optimization algorithm. Assume a population with \( N \) particles. The position of each particle \( i \) at any time is a \( n \)-dimensional vector in decision space. In the \( t \) iterations, the position of the particle \( i \) is denoted as \( x_i(t) = [x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t)]^T, (i = 1, 2, \ldots, N) \). The adaptive value of each particle can be obtained from \( x_i(t) \) substituted to the objective function or the fitness function. The particle is judged by the fitness value [2, 19]. Personal best of the particle \( i \) in the \( t \) iteration is denoted as \( X_i^{pb}(t) = \)
\[
[X_{i1}^{pb}(t), X_{i2}^{pb}(t), \ldots, X_{in}^{pb}(t)]^T. \text{ Global best of the particle } i \text{ in the } t \text{ iteration is denoted as } X_i^{gb}(t) = [X_{i1}^{gb}(t), X_{i2}^{gb}(t), \ldots, X_{in}^{gb}(t)]^T. \text{ The velocity of the particle } i \text{ in the } t \text{ iteration is denoted as } v_i(t) = [v_{i1}(t), v_{i2}(t), \ldots, v_{in}(t)]^T, (i = 1, 2, \ldots, N).
\]

In the \(t + 1\) iterations, the position \(x_i(t + 1)\) and the velocity \(v_i(t + 1)\) of the particle \(i\) are updated with

\[
v_i(t + 1) = wv_i(t) + c_1r_{i1}(X_{id}^{pb}(t) - x_{id}(t)) + c_2r_{i2}(X_{id}^{gb}(t) - x_{id}(t)) \tag{10}
\]

\[
\begin{align*}
    v_{id} &= v_{id}^{\max}, v_{id} \succ v_{id}^{\max} \\
    v_{id} &= v_{id}^{\max}, v_{id} \prec -v_{id}^{\max} \\
    v &= [v_1^{\max}, v_2^{\max}, \ldots, v_n^{\max}]^T
\end{align*} \tag{11}
\]

where \(v^{\max}\) is multi-dimensional limit velocity, usually taken as constant vector. \(c_1\) and \(c_2\) are learning factor, which is nonnegative constant, \(r_{i1}\) and \(r_{i2}\) is the random number between 0 and 1, \(w\) is expressed as

\[
w = w^{\max} - \frac{(w^{\max} - w^{\min}) \times t}{T^{\max}} \tag{12}
\]

where \(w^{\max}\) is the maximum value, \(w^{\min}\) is the minimum value, \(t\) is the current number of iterations, \(T^{\max}\) is the maximum number of iterations. The inertia weight is linearly declining \([8]\).

Personal best \(X_i^{pb}(t + 1)\) of the particle \(i\) in the \(t + 1\) iteration is updated with

\[
X_i^{pb}(t + 1) = \begin{cases} 
    x_{id}(t + 1), F(x_{id}(t + 1)) \prec F(X_i^{pb}(t)) \\
    x_{id}(t), F(x_{id}(t + 1)) \geq F(X_i^{pb}(t))
\end{cases} \tag{13}
\]

where \(F(x)\) is fitness function.

Personal best \(X_i^{gb}(t + 1)\) in the \(t + 1\) iteration is selected by using

\[
X_i^{gb}(t + 1) = \text{arg} \left\{ \min F(X_i^{pb}(t + 1)) \right\} \tag{14}
\]

where \(\text{arg}\) is the value of argument \(x\) of \(F(x)\) \([10]\).

By using the solving method of fitness, the particle swarm is initialized randomly in the feasible domain, and the particle with non-dominated solution is selected to construct the elitism set \(NP_i\). The particle in the elitism set can guide the flight of other particles during the population renewal process. The fitness of elitism particle is solved by using niche strategy. The higher the aggregation of the particles, the smaller the fitness of the particles. The fitness of the \(i\) th particle is defined as

\[
F(i) = 1/m_i \tag{15}
\]

where \(m_i\) is the niche number of the personal particles.

\[
m_i = \sum_{j=1}^{N} \text{sh}(d(i, j)), i = 1, 2, \ldots, N \tag{16}
\]

where \(\text{sh}(d(i, j))\) is fitness sharing function between the particles \(i, j\).

\[
\text{sh}(d(i, j)) = \begin{cases} 
    1 - \left( \frac{d(i, j)}{\sigma_{\text{share}}} \right)^\alpha, d(i, j)/\sigma_{\text{share}} \\
    0, \text{others}
\end{cases} \tag{17}
\]
where \( d(i, j) \) is the distance of the particle \( i \) and \( j \), \( \sigma_{\text{share}} \) is the niche radius, taken as constant \( \alpha \) is the parameter of control the shape of \( sh(d(i, j)) \), usually set to \( \alpha = 1 \).

\( \sigma_{\text{share}} \) can also be dynamically adjusted by

\[
\sigma_{\text{share}} = N \frac{1}{m} \cdot \frac{d^{(n)}}{2}
\]  

where \( n^{(n)} \) is the diameter of the \( n \)th generation of hypersphere, decided by the isotactic surface formed with non-dominated solutions, \( m \) is the number of the objective function, \( N \) is the size of population.

Roulette method is used to select the global best according to the fitness of each particle in the elitism set. In the operation of the algorithm, the particle with the best position guides other particles to update their position and velocity according to Eq. (10). After iteration, the non-dominated solution of the obtained particle is added to the elitism set, so that the number of elitism particles will increase as the number of iterations increases, which will affect the operation speed of the algorithm. Therefore, there is a need to limit the capacity of the elitism set. When the number of elitism particles exceeds their maximum capacity, the fitness of each particle in the elitism set is calculated according to Eq. (15), and the particles with smaller fitness are deleted. When the algorithm satisfies the terminate iteration condition, the particles in the elitism set can be considered as the Pareto optimal solution.

The new algorithm improves the maximum velocity limit mode. In the process of solving the limited condition (such as the capacity of the elitism set \( M \) in step (1)), the small probability variation mechanism is improved in step (6). The steps of the algorithm are as follows.

1. Set population size \( N \), maximum velocity \( v^{\text{max}} \), the capacity of the elitism set \( M = \phi \) and number of iterations \( t = 0 \). The condition of the termination of the algorithm is set as the number of iterations reaches the preset value. The location and velocity of the particle is initialized randomly, and the fitness value of each particle is calculated, and the initial solution is obtained.
2. The non-dominated solution is added to the elitism set. By using Eq. (15), the fitness of each particle is solved.
3. When the number of iterations is less than the preset value, step (4) \( \sim \) (7) are executed, or end the algorithm.
4. The global best \( X_{i}^{pb} \) is selected according to the fitness of each particle in the elitism set. The position and velocity of each particle is updated according to Eq. (10), the fitness of each particle in the population is calculated again.
5. Repeat step (2). In order to ensure that the number of individuals in the elitism set is not more than its capacity, the dominated solution with small fitness is deleted.
6. If the position of the current particle is superior to its historical location \( X_{i}^{pb} \), \( X_{i}^{pb} \) is replaced by the current particle position. If the current particle is neither superior nor inferior to its personal best, then it will keep or replace the particle inferior to \( X_{i}^{pb} \) according to the probability of 50\% , and make variation according to the 5\% probability, that is, to initialize the particle again.
(7) If the current position of a particle is superior to the global best of all the particles $X_{gb}^i$, it is updated, otherwise the best position of the whole particle $X_{gb}^i$ will remain unchanged.

(8) The finally obtained elitism set is the Pareto optimal solution set.

2.3. Multi-scale variation particle swarm optimization. In order to prevent the algorithm from falling into the local best, if the position of the particle is mutated, the particle can be located to other area of the solution space, thereby increasing the probability of finding the global best solution. However, if a single uniform variation scale is used to help the algorithm escape from the local best, though it can escape from the origin, but the distance between the local best values cannot be predicted in advance, so the appropriate variation scale cannot be given. If the scale is too large, the fitness of the new position after the variation is not necessarily better than the existing optimal solution. Especially in the late evolution of the algorithm, the optimal solution may exist in the area around the existing optimal solution. In this way, the best position cannot be determined by the simple uniform variation, and it cannot converge to the global best solution because of the limit of the number of iterations. In order to make up for the lack of uniform variation operation, a Gaussian variation method with different scales is introduced \[12,18\].

Considering that the position of the particle determines the scale of the variation \[14\], the specific design process of the Gaussian variation operator is as follows. First, a variable is defined to evaluate the current position of a particle. In the single objective optimization problem, the objective function value is used, that is, the fitness function value to evaluate the location of the particle, but multiple objective functions to be optimized must be considered for multi-objective optimization problems, without a fixed form of fitness function. The fitness value is defined as follows.

$m$-dimensional objective space is divided into $K_1 \times K_2 \times \cdots \times K_m$ elements. $K_i (i = 1, 2, \ldots, m)$ is the number of the elements for each dimensional objective space. Its value is adjusted according to the specific problem. Each component $d_i$ of the unit width is given by

$$d_i = \frac{\max_{x \in D} f_i(x) - \min_{x \in D} f_i(x)}{K_i} \tag{19}$$

Let $F_{i}^{\max} = \max_{x \in D} f_i(x), F_{i}^{\min} = \min_{x \in D} f_i(x)$, and $D$ is the decision space.

Assume $c(F_1^{\min}, F_2^{\min}, \ldots, F_M^{\min})$ is the origin of the objective space under the current number of iterations, that is, intersection point of all lower bounds of the objective space. For the particle $s$, the coordinates of its corresponding objective vector are $(s_1, s_2, \ldots, s_m)$, then the distance of the point $c$ and the point $s$ is $(t_1, t_2, \ldots, t_M)$, where $t_i = s_i - F_{i}^{\min}, i = 1, 2, \ldots, M$. Then home address of the point $s$ in the $i$th dimension is $h_i = \text{mod}(t_i d_i) +$, where $\text{mod}(t_i, d_i)$ is integral part of $t_i/d_i$. The particles with the same objective vector of home address belong to the same element, and the particles with the same element are the family member of the element.

After the objective space is divided into many elements, two indexes are designed to determine the fitness value of the individual particle. These two indexes are health index and congestion index. The health index describes the number of other particles dominated by a particle in iteration, and the congestion index describes
the number of family members in the element of the corresponding to the objective vector of a particle. The fitness value of the \( k \)th evolutionary particle is defined by

\[
\text{fit}(c, k) = \frac{\text{density}(c, k)}{H(c, k) + 1} \tag{20}
\]

where \( \text{density}(\cdot) \) is the number of the particles of the element of the \( k \) generation particle \( c \), \( H(\cdot) \) is the dominated particles of the \( k \) generation particle \( c \). The healthier a particle is, the more the number of other particles it dominates, the better its fitness is. The less the number of family members contained in the particle, the better the fitness of the particle. The purpose of this definition is to obtain close to optimal, evenly distributed, and well extended Pareto front-end.

The design method of multi-scale variation operator is given as follows. Assume \( W \) is the number of the scales. First the initial assignment of the standard deviation of the multi-scale Gaussian variation operator is given by

\[
\sigma^{(0)} = (\sigma^{(0)}_1, \sigma^{(0)}_2, \ldots, \sigma^{(0)}_W) \tag{21}
\]

Generally, the initial standard deviation is assigned to the value interval of the decision variable. The value of \( \sigma \) is constantly adjusted by the following method. If the number of the particles is \( Z \), the particles are sorted in the ascending order according to the fitness value and then combined into \( W \) subgroups in sequence. The number of the particles of each subgroup is \( Q = Z/W \). \( k \) is the number of the current iteration. The fitness of each subgroup is calculated by

\[
\text{fit}X^{(k)}_w = \sum_{i=1}^{Q} f(x^{w}_i)/Q, w = 1, 2, \ldots, W \tag{22}
\]

where \( f \) is the fitness value. After competing of individuals with each other, the adaptation ability is the basis for the different variation ability of the particle. Therefore, the standard deviation of the \( W \) variation operator is given by

\[
\sigma^{(k)}_w = \sigma^{(k-1)}_w \ast \left( 1 - \frac{k}{K_{\text{max}}} \right) \ast \left| \frac{W \ast \text{fit}X^{(k)}_w}{\text{fit}_{\text{max}} - \text{fit}_{\text{min}}} \right| \tag{23}
\]

where \( k \) is the number of the current evolutional generation, \( K_{\text{max}} \) is the maximum number of the evolutional generation. In order to prevent the value of the variation operator beyond the scope of the solution space, if \( \sigma^{(k)}_i > B \) is the width of the solution space, the iteration is terminated until \( \sigma^{(k)}_i > B \) is satisfied.

\[
\sigma^{(k)}_i = \left| B - \sigma^{(k)}_i \right| \tag{24}
\]

In particle swarm optimization, the position and velocity of the particle are updated according to Eq. (10), so that they can move to the personal best and the global best through continuous self-learning behavior and other excellent individual experience. In the multi-objective optimization problem, there are many solutions that cannot be compared with each other. In order to ensure the non-dominated solution set with uniform distribution, the following improvements are made to the Eq. (10) for the multi-objective optimization problem.

\[
v^{k+1}_i = \nu^k_i + c_1 r_1 (pbest^k_i - x^k_i) + c_2 r_2 (gbest^k_i - x^k_i) \tag{25}
\]

While ensuring that particles are approaching to the personal best and the global best, the particle is guided to fly to the relatively sparse area of the objective space,
so that the non-dominated set will not be concentrated in one area, and the non-dominated set will be distributed uniformly and expand well.

The solution of the multi-objective optimization problem is usually not unique, and the group produces a number of non-dominated solutions in each generation. In this paper, the elitism archiving strategy is adopted to construct an optimal solution set (non-dominated solution set) to store the optimal solution obtained by each iteration. The member of this set is called non-inferior solution or Pareto optimal solutions.

How to select the personal best value and the global best value is one of the main problems to be solved when the basic particle swarm optimization (PSO) algorithm is used to deal with the multi-objective optimization problem. In the single objective PSO, the global best $g_{best}$ of each particle is usually the same, and each time the best solution is only required to be defined as the global best position. In the multi-objective optimization process, the global best point is no longer a single solution, but a set of non-inferior solutions. The key of the problem is how to select the global best points for each particle to obtain a number of Pareto solutions which are distributed widely and evenly. In this paper, selection of the global best method is as follows: when the objective space is divided into many elements, the fitness value is defined for the element containing at least one non-inferior solution. Then for each particle, a lattice is selected according to the roulette method and a non-inferior solution is selected randomly as $g_{best}$ of the particle.

The selection method of personal best is as follows. First, the domination relationship of the position of the particle $x$ and the last generation of the personal best is judged. If $g_{best}$ dominates the particle $x$, then the personal best of $x$ is unchanged. If $x$ dominates $p_{best}$, $p_{best}$ is replaced by $x$ as the new personal best. If $p_{best}$ and $x$ do not dominate each other, the previous algorithm keeps $p_{best}$ unchanged or randomly selects one as a new personal best with the probability of 0.5. In the proposed algorithm, the selection probability of $x$ and $p_{best}$ is adjusted, which is to select the probability of $p_{best}$ is 0.2 and the probability of $x$ is 0.8. In this way, it is possible to prevent a number of successive iterations of $p_{best}$ without change, and the particle is returned to the previously searched area by the effect of $p_{best}$. Therefore, the personal best updating method is more conducive to maintaining the diversity of the population.

In order to explain the optimization mechanism of this algorithm more effectively, the problem to be optimized contains only one objective function $f(x)$, as shown in Fig. 1. One of the particles of a population has evolved to the $a$ point after multiple

![Figure 1. Optimization mechanism of multi-scale variation](image-url)
Figure 2. Improved particle swarm optimization algorithm for multi-objective optimization problem

evolutions. After small scale and deep search in variation operation, the particle position $a'$ is found down the hill and then evolve from algorithm to $a''$. After large scale variation operation, the particle position $b$ with better fitness value is found, and then the particle $b$ is selected to enter the next generation. After the algorithm itself evolved several and small scale variations, the particle with the position $b'$ is found. After large scale variation of the particle $b''$, the particle $c$ is found. Then $c$ is taken as the particle for the next generation of evolution. Finally, the optimal solution $c'$ is found by self - evolution and small scale variation.

According to the above analysis, assuming that the whole decision space is a valley, the operation of large scale variation is to go down the hill to find a better position than the present one ($a'' \rightarrow b, b' \rightarrow c$). Large scale variation operation is a rough search behavior, which is used to search decision space in a wide range. If the large scale variation operation finds a better region of fitness, the evolutionary operation and small scale variation operation of the algorithm can deep excavate near the area through local search to find a more accurate solution ($a \rightarrow a', a' \rightarrow a'', b \rightarrow b', c \rightarrow c'$). The exploration of the region near the optimal solution is accomplished by the evolution of the algorithm itself and the operation of the small scale variation.

The proposed multi-scale variation particle swarm optimization algorithm is used to solve the multi-objective optimization problem, as shown in Fig. 2.

1. The velocity and position of the individual in the population $P$ is initialized, and the objective vector value of each individual is calculated.
2. The individuals that are not dominated by any other particle in the particle swarm are kept in the non-inferior solution set, and the positions of these particles are non-inferior solutions.
3. pbest of each particle is determined.
4. The target space is divided into many elements, and the element is determined according to the objective vector corresponding to the particle.
5. According to Eq. (20), the fitness value is defined for each element. For each particle, gbest is selected by the global best selection method.
6. The velocity and position of the particle is updated according to the Eq. (25).
7. The motion range of particles from crossing the boundaries of decision space is prevented according to the following ways: if the position of the particle...
exceeds the range of a decision variable, the position of the particle is the boundary value, and the velocity of the particle is set to opposite to that before.

(8) The standard variance of the variation operator is calculated, and the multi-scale variation operation is carried out for each particle position.

(9) The mutated particles are recombined with the original population into a new population, and the non-inferior sets are updated and maintained on the basis of the objective vector of each particle.

(10) The search is stopped when the terminating condition is satisfied, otherwise, turn (3) to continue to iterate.

3. Experimental results and analysis. In order to analyze the global search performance and convergence speed of the multi-scale variation particle swarm optimization algorithm, a simulation environment is established. Test hardware is AMD Phenom II X4 B95 3.0 GHz and 2G RAM, and the software is Microsoft Windows XP and MATLAB R2017b. The test data is randomly generated. The amount of tasks is limited to 1000 to 100 billion instructions, and the computing performance of each data center is set to (500, 1000, 2000, 4000). The bandwidth between data centers is set to 100∼10000MB. The size of data transmission between tasks is set to 100∼10000MB. The density \( d \) of data transmission between tasks is obtained with Eq. (26), where \( n \) is the total number of tasks, \( |E| \) is the number of times all tasks need to be communicated. The algorithm for improved the mutation of PSO is used for test. The introduction of 4 Benchmark optimization problems is for analysis. The algorithm parameters are set as follows. \( w \) is 0.5, \( c_1 \) and \( c_2 \) is 1.4. In the proposed algorithm, \( M = 3 \), initial standard deviation \( \sigma_0 \) is the range of optimization variable. The population size of all the tests is 50, the function dimension is set to 30, and each time runs 3000 generations.

\[
d = \frac{|E|}{n(n - 1)/2} \tag{26}
\]

In order to verify the solution quality of the algorithm for non-inferiority, the non-inferior solution of the proposed algorithm is compared with strength Pareto evolutionary algorithm (SPEA2), non-dominated sorting genetic algorithm (NSGAII), several common multi-objective particle swarm algorithm (CMOPSO) and crowding distance sorting based particle swarm algorithm (CDMOPSO). The parameter setting of the algorithms is as shown in Table 1.

### Table 1. Experimental parameter setting

| Algorithm    | Number of particles | Size of non-inferior solutions | Number of iterations | Probability of intersecting | Probability of variation | Size of the real solution set |
|--------------|---------------------|--------------------------------|----------------------|-----------------------------|-------------------------|-----------------------------|
| NSGAII       | 160                 | 50                             | 100                  | 0.9                         | 0.1                     | 500                         |
| CMOPSO       | 160                 | 50                             | 100                  | Nonlinear decline           |                         | 500                         |
| SPEA2        | 160                 | 50                             | 100                  | 1                           | 1/n                     | 500                         |
| CDMOPSO      | 160                 | 50                             | 100                  | 0.5                         |                         | 500                         |
| The proposed algorithm | 160     | 50                             | 100                  | Decrease with the increase of k |                         | 500                         |
In the experiment, taking the shortest total completion time as the aim, the situation of different number of iterations is tested when the improved particle swarm algorithm is different in the number of tasks. The size of the task during the experimental test is randomly generated by a function calculation between $[500MI, 5000MI]$. Each case runs randomly for 100 times and the average value is obtained. Experimental results are shown in Fig. 3 and Fig. 4.

From comparison of the improved multi-scale particle swarm algorithm, traditional particle swarm algorithm, and hybrid disturbance particle swarm algorithm, it can be obtained that the early iteration has faster convergence speed, but in the late iteration of the algorithm, the improved algorithm convergence results better, and more stable, the total completion time is shorter and greatly meets the need of the user.

Fig. 5 and Fig. 6 show the comparison of the convergence of the DTLZ test function and the uniformity of the solution set between the improved particle swarm optimization and the traditional particle swarm optimization algorithm.

From the Fig. 5 and Fig. 6, it can be obtained that the traditional particle swarm optimization algorithm cannot converge to the front of the correct Pareto for DTLZ2 optimization. Although the proposed algorithm has more convergence times than other functions, it still converges successfully, and the uniformity of the solution is good. The test functions of other targets can be correctly converged to the global best Pareto surface, and the individuals in the solution set is approximates with
isometric distribution. The proposed algorithm has a significant improvement in the performance of the test function, and the improved particle updating equation obviously improves the convergence speed of the algorithm.

4. Conclusions. Aiming at the shortcomings of traditional particle swarm optimization algorithm, such as slow convergence speed and easy to fall into local best value, this paper proposes an improved particle swarm optimization algorithm based on multi-scale variation. The algorithm uses the Gaussian variation mechanism of different scales to escape the local solution. In the early stage of algorithm search, the large scale variation operator can complete the general location of the best solution area while the variation scale gradually decreases. Finally, in the later stage of algorithm search, the small scale variation operator is applied to achieve the deep mining of the area near the optimal solution. A new speed updating formula is adopted to maintain the diversity of the solution for the multi-objective optimization problem. The convergence mechanism of the algorithm is also analyzed.

There are still many shortcomings in this paper to be further improved.

(1) Based on the improvement of the optimization algorithm for solving multi-objective optimization problem, there are a lot of problems to be solved for the complexity of multi-objective optimization problem. Therefore, how to improve multi-objective optimization problem is the next subject to be researched.
(2) The research in this paper is also limited to the experimental simulation of the artificial structure test problem, and has not been applied to practical application. The next step is to model and optimize the practical application problems.

REFERENCES

[1] Z. Beheshti, S. M. Shamsuddin and S. Hasan, Memetic binary particle swarm optimization for discrete optimization problems, *Information Sciences*, 58–84.

[2] M. C. and Q. L., Multiobjective optimization of switched reluctance motors based on design of experiments and particle swarm optimization, *Eurasia Journal of Mathematics Science & Technology Education*, 3 (2017), 1144–1153.

[3] G. H. Chen, Y. Zhao and B. Su, Raw material inventory optimization for mto enterprises under price fluctuations, *Journal of Discrete Mathematical Sciences & Cryptography*, 20 (2017), 255–270.

[4] H. L. Chen, B. Yang, S. J. Wang, G. Wang, D. Y. Liu, H. Z. Li and W. B. Liu, Dynamic multi-objective particle swarm optimization based on projection mapping, *Computer Simulation*, 233–238.

[5] R. Cheng and Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, *Information Sciences*, 291 (2015), 43–60.

[6] J. M. Feng and S. Y. Liu, Particle swarm optimization algorithm based on inertia weight exponentially decreasing for solving absolute value equations, *Journal of Jilin University (Science Edition)*, 54 (2016), 1265–1269.

[7] M. Hajihassani, D. J. Armaghani, M. Monjezi, E. T. Mohamad and A. Marto, Blast-induced air and ground vibration prediction: A particle swarm optimization-based artificial neural network approach, *Environmental Earth Sciences*, 74 (2015), 2799–2817.

[8] X. He, H. Guan and J. Qin, A hybrid wavelet neural network model with mutual information and particle swarm optimization for forecasting monthly rainfall, *Journal of Hydrology*, 17 (2015), 88–100.

[9] W. Hu and G. G. Yen, Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system, *IEEE Transactions on Evolutionary Computation*, 19 (2015), 1–18.

[10] G. P. Karatzas and Z. Dokou, Optimal management of saltwater intrusion in the coastal aquifer of malla, crete (greece), using particle swarm optimization, *Hydrogeology Journal*, 23 (2015), 1181–1194.

[11] C. Lee, Assigning the appropriate works for review on networked peer assessment, *Eurasia Journal of Mathematics Science and Technology Education*, 3, 3283–3300.

[12] B. M. R. and M. Z., Analysis of stability, local convergence, and transformation sensitivity of a variant of the particle swarm optimization algorithm, in *IEEE Transactions on Evolutionary Computation*, 370–385.

[13] L. Mao, W. Guiwu, F. Alsaaadi, T. Hayat and A. Alsaedi, Hesitant pythagorean fuzzy hamacher aggregation operators and their application to multiple attribute decision making, *Iran. J. Fuzzy Syst.*, 13 (2016), 1–16, 147.

[14] S. L. Marie-Sainte, A survey of particle swarm optimization techniques for solving university examination timetabling problem, *Artificial Intelligence Review*, 44 (2015), 537–546.

[15] M. Pelto, Maximum difference about the size of optimal identifying codes in graphs differing by one vertex, *Discrete Mathematics & Theoretical Computer Science*, 17 (2015), 339–356.

[16] C. S., M. J. and B.-R. A., *Measuring the Curse of Dimensionality and Its Effects on Particle Swarm Optimization and Differential Evolution*, 3, Applied Intelligence, 2015.

[17] J. Sun, X. Wu, V. Palade, W. Fang and Y. Shi, Random drift particle swarm optimization algorithm: convergence analysis and parameter selection, *Machine Learning*, 101 (2015), 345–376.

[18] S. V., P. S. K., V. J. and et al, Particle swarm optimization inversion of self-potential data for depth estimation of coal fires over east basuria colliery, jharia coalfield, india, *Environmental Earth Sciences*, 1–12.

[19] X. Wang, J. Wu and L. Liu, The development of modern service industry in china and its influence factors, *Journal of Interdisciplinary Mathematics*, 20 (2008), 161–171.

[20] L. Y., Z. Z.H., L. S. and et al, Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems, *European Journal of Operational Research*, 3 (2015), 370–382.
[21] H. Q. Yan, Network intrusion small signal detection model based on optimization particle swarm algorithm, *Bulletin of Science & Technology*, 12 (2015), 193–195.

[22] R. Yan, M. A. Xiao-Juan, T. Z. Lian and L. Wang, Research on optimal joint problem of routing and loading in military airlift, *Journal of China Academy of Electronics & Information Technology*, 7–13.

[23] B. Yao, B. Yu, P. Hu, J. Gao and M. Zhang, An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot, *Annals of Operations Research*, 242 (2016), 303–320.

Received June 2017; revised November 2017.

E-mail address: zhangmin_1233@163.com
E-mail address: lg55071406@126.com