A Method of Searching for Optimal Coalition Structure for Solving Resource Scheduling Problem of Overall Load Balancing in Edge Computing Environments

Hu Yanan¹, Li Chunsheng¹²*, Zhang Kejia¹²

¹School of Computer & Information Technology, Northeast Petroleum University, Daqing 163318 (PR China)
²Heilongjiang Provincial Key Laboratory of Oil Big data & Intelligent Analysis, Daqing 163318 (PR China)
*Corresponding author’s e-mail: slee@nepu.edu.cn

Abstract. The cooperative computing resource scheduling problem in the distributed computing environment is usually transformed into the problem about how to find the optimal coalition structure. However, affected by factors such as high node density and large number of concurrent tasks in edge computing environments, it is difficult to formulate resource scheduling schemes for overall load balancing in a large strategic space by traditional methods. In order to solve this problem, we proposed a method of searching for optimal coalition structure based on discrete particle swarm optimization (DPSO). Firstly, we converted the resource scheduling problem of edge computing into an optimization problem model, constructed a new structure of coalition structure to express the resource scheduling scheme, and gave key elements such as optimization objectives, decision variables, and constraint conditions. Secondly, we improved DPSO and designed M-ary discrete particle swarm optimization (MDPSO): improved particle state determination method for adapting to the new expression of the coalition structure, and proposed a power series-based inertia weight computing method to improve the optimization speed and the optimal solution quality. Finally, we simulated real application scenarios and designed comparative experiments on the optimization speed, the optimal solution quality, stability, etc., which verifies that the method proposed in this paper has obvious advantages in solving the resource scheduling problem of overall load balancing in edge computing.

1. Introduction
Coalition is a collection of agents generated to accomplish specific tasks [1]. It forms with the arrival of new tasks, and disintegrates with the completion of tasks [2]. The coalition structure is a collective term of all coalitions in multitask concurrency, which can abstractly describe the strategies adopted by agents to initiate cooperative behaviors [3]. Therefore, the study of “how to solve the optimal coalition structure” can be used to solve practical problems such as resource scheduling in the cloud computing environment [4], robot configuration scheme design [5], and project R&D team structure optimization [6].

In recent years, the edge computing technology has ushered in new challenges while developing rapidly, one of which is the problem of resource scheduling oriented to overall load balancing [7]. Relevant research achievements have successfully solved the resource scheduling problems in
different scenarios such as cloud-free cantor [8], multi-objective [9], and diverse tasks [10], but in the scenario with high member density and a large number of concurrent tasks, they always face the dilemma of large strategic space, low search speed and low quality of optimal solution.

Another way is study on the game algorithm [11], ultimately introducing methods from economics and game theory while solving similar problems [12-13]. In the meantime, many research results [14-15] have shown that “facing overall load balancing” means higher optimal solution quality; at the same time, the huge strategic space will also restrict the search speed of the optimal solution. Therefore, based on the previous research work [16], and with reference to the idea of Heap Intelligent Discrete Particle Swarm Optimization (HIDPSO) to solve the problem of optimal resource allocation [17], we constructed an expression of the coalition structure which can describe the resource scheduling scheme concisely and clearly, based on which an optimal coalition structure solution method was proposed for the overall load balancing in edge computing under a large strategic space.

2. Problem Description

2.1 Variable Definition

Definition of Scenario H: The number of members on the edge side is C, and the number of concurrent tasks is L. \( U^n \) represents the set of coalitions that may be formed to complete the task \( T_n \), any element can be expressed as \( U^n = (N^n, v^n) \), \( N^n \) represents the number of members participating in \( U^n \), and \( v^n \) is the utility of the coalition \( U^n \). The basic concepts, formulas, and reasoning processes can be described as follows.

Utility function: We define that \( r_m \) stands for the rewards after completing \( T_n \), \( c^n_k \) stands for the computing power expenditure of \( U^n \), and \( e^n_k \) the additional expenditure of \( U^n \); the expression of \( v^n \) given by “utility = reward - cost” is shown in the formula (1).

\[
v^n = r_m - c^n_k - e^n_k
\] (1)

Additional expenditure refers to the expenditure among coalition members, including communication loss, computing power redundancy, etc. We define \( e^n_k = E(N^n) \), and then \( E(N^n) \) must be the monotonic increasing function. \( p_i \) is defined as the computing power of the member \( S_i \), and \( t^n_{i,k} \) the time consumed by \( S_i \) participating in \( U^n \), then we obtain the computing power expenditure \( c^n_{i,k} = F(p, t^n_{i,k}) \) [18]. By updating the formula (1), the utility function is obtained, as shown in the formula (2).

\[
v^n = r_m - \sum_{k=1}^{N^n} F(p, t^n_{i,k}) - E(N^n)
\] (2)

Cost performance function: All members wish to join the coalition that maximizes their utility. This is a manifestation of the “individual rationality” of the member, and this intention is usually reflected with cost performance [19]; the cost performance of \( U^n \) is \( b^n = v^n / t^n \), where, \( t^n \) denotes the expected completion time of \( U^n \); According to the barrel principle, \( t^n \) must depend on the member consuming the longest time in \( U^n \), that is, \( t^n = \max_{i \in N^n} (t^n_{i,k}) \); the expression of the updated cost performance \( b^n \) is shown in the formula (3).

\[
b^n = \frac{r_m - \sum_{k=1}^{N^n} F(p, t^n_{i,k}) - E(N^n)}{\max_{i \in N^n} (t^n_{i,k})}
\] (3)

Coalition structure: In Scenario H, any possible coalition structure \( M_j (M_j \in M) \) consists of \( L \) coalitions. The traditional expression of the coalition structure usually comprises the figure [10], matrix [11] or unequal linked list [20], which has problems such as trivial structure and complicated calculation process when the strategic space is large. Therefore, a new expression of coalition structure
was constructed: we define that $B_i$ denotes the task $B_i \in [0, L]$ that $S_i$ participates in, and then $M_j$ can be written as shown in the formula (4).

$$M_j = \langle B_{i_j}, B_{i_j} \rangle \ldots B_{i_j} \rangle$$

(4)

Where, $B_0 = 0$ means that $S_i$ does not process any task. For example, in a scenario where six members complete three tasks, $M = \langle 1, 3, 2, 1, 3, 0 \rangle$ represents that $T_j$ is completed by the coalition consisting of $S_i$ and $S_j$, $T_2$ by the coalition consisting of $S_j$, and $T_3$ by the coalition consisting of $S_1$ and $S_3$, while $S_5$ does not process any task. According to the formula (4), the strategic space size constituted by $M_j$ is $W = \text{count}(M_j) = (L + 1)^y$.

### 2.2 Problem Model

The problem model was established based on Scenario $H$, where the decision variable is $M_j$, the optimization objective and constraint conditions are described as below.

**Optimization objective:** The “optimal” coalition structure is to try to satisfy the supplier requirements on the premise of giving priority to the customer expectations. “Customer expectations” refer to “the shortest possible completion time”, and is recorded as $\text{Arg min}(f(M_j))$; the value depends on the last member completing the task, that is, $f(M_j) = \max_{j \in C} (t'_j)$. Since the member’s expected completion time is inversely proportional to its own computing power, “the shortest possible completion time” is equivalent to “the greatest possible utilization of computing power resources” and can be described by the formula (5), where, $t'_j$ indicates the expected completion time of the member $S_i$ in $M_j$.

$$\text{Arg min}(f(M_j)) = \text{Arg max} \left( \frac{1}{\max_{n \in C}(t'_j)} \right)$$

(5)

“Supplier requirements” include “the lowest possible cost expenditure” and “the highest possible cost performance”. In a non-superadditive environment, “the lowest possible cost expenditure” is equivalent to “the highest possible total utility”. The utility objective function of the coalition structure $M_j$ is defined as $v(M_j)$, and then “the highest possible total utility” can be expressed as $\text{Arg max}(v(M_j))$. The expression given according to the formula (2) is shown in the formula (6), where, $v^n_j$ stands for the utility of the $m$-th coalition constituting $M_j$.

$$v(M_j) = \sum_{n=1}^{L} v^n_j = \sum_{n=1}^{L} r_n - \sum_{n=1}^{L} \sum_{i \in C} F(p_i, t^{n}_{j}) - \sum_{n=1}^{L} E(N^n)$$

(6)

$b(M_j)$ is defined as the cost performance objective function of the coalition structure, and then “the highest possible cost performance” can be expressed as $\text{Arg max}(b(M_j))$. Considering the stability of coalition structure, “the highest possible cost performance” requires the minimum cost performance difference among the coalitions that constitute the coalition structure. Standard deviation is introduced to represent the cost performance difference $[21]$, and the expression of $b(M_j)$ obtained is as shown in the formula (7), where, $b^n_j$ is expressed by the formula (3).

$$b(M_j) = \sum_{m=1}^{L} \left( b^n_j - \frac{\sum_{m=1}^{L} b^n_j}{L} \right)^2$$

(7)

**Constraint conditions:** The constraint conditions are described as follows by the definition of Scenario $H$: (1) The number of members of $M_j$ must be larger than $C$; (2) Each member can only participate in completing one task ($B_1$ is the unique value), that is, any coalition $U_1, U_2$ constituting $M_j$ must satisfy $U_1 \cap U_2 = \emptyset$; (3) Each task requires at least one member to participate in; (4) The
expected completion time must be less than the planned completion time, that is, $f(M_j) < t$, where, $t$ is the expected completion time of $T$.

$$f(M_j) = \max_{j \in C}\left(\frac{\text{datasize}}{\text{bandwidth}} + \frac{\text{computation}}{p_i}\right)$$

(8)

3. Search for the Optimal Coalition Structure

3.1 Fitness Function and Encoding Method

The normalization of the objective function is usually needed to solve the multi-objective optimization problem. In this paper, the linear weighted sum method was used to transform the multi-objective function into a single objective function, while the method to process the objective function in different scenarios can be designed as needed. We define the weight coefficient $w = \{w_j, w_i, w_k\}$, and it satisfies $w_j + w_i + w_k = 1$. According to the formula (5), (6) and (7), the expression of $P(M_j)$ is written as the formula (9).

$$P(M_j) = w_j \times \text{Arg max} \left(\frac{1}{f(M_j)} + w_i \times \text{Arg max}(v(M_j)) + w_k \times \text{Arg max}(b(M_j))\right)$$

(9)

According to the formula (4), a simplified encoding method of coalition structure is given: $M_j = B_1B_2B_3...B_i...B_c$, $\forall B_i \in [0, L]$ . This encoding method can be considered as the $L+1$ decimal number(s) of length $C$.

3.2 MDPSO and Computation Complexity Analysis

We adopted the DPSO and made improvements to the problem model: To adapt to the expression of the coalition structure in the formula (4), we improved the particle state determination method based on the sigmoid function; to avoid “missing the optimal solution” and improve the search speed, we proposed a power series-based inertia weight computing method.

Taking the basic particle swarm optimization (PSO) as an example, the parameters are defined as follows: the number of strategies is $W$, the dimensionality $C$, the number of particles $K$, and the number of iterations $D$; the flight speed of the particle $u$ ($u < K$) at the $d$-th time ($d < D$) is $V_u^d$, the position $Z_u^d$, and the fitness function $P(Z_u^d)$; $P_{\text{best}}$ stands for the historically optimal solution of the particle $u$, and $P_{\text{g-best}}$ the historically optimal solution of all particles. According to the definition of basic PSO, the speed update and position update formulas of the particle $u$ are given as shown in the formula (10) and (11).

$$V_u^{d+1} = \omega \times V_u^d + c_1 \times q_1 \times (P_{\text{best}} - P(Z_u^d)) + c_2 \times q_2 \times (P_{\text{g-best}} - P(Z_u^d))$$

(10)

$$Z_{u,d+1} = Z_{u,d} + V_u^{d+1}\quad (11)$$

$c_1$, $c_2$ represent the learning factor, reflecting the self-summarizing ability and social learning ability of the particle. $q_1$, $q_2$ are the random number, and the value range is [0, 1]. $\omega$ represents the inertia weight, used to equalize the local and global optimization. Generally, the inertia weight is calculated by linearly decreasing with the number of iterations, that is, the inertia weight at the $d$-th iteration is $\omega(d) = \omega_{\text{max}} - \frac{\omega_{\text{max}} - \omega_{\text{min}}}{D} \times d$. At the beginning of the iteration, this computing method will cause the decline rate of inertia weight (that is, acceleration $a = d/D$) to change too fast, resulting in that it is to easily “jump” over the optimal solution in the high-density strategic space; and when the iteration is about to end, the acceleration changes very slowly, which will affect the search speed in a larger strategic space. Therefore, we introduced the parameter $p$ to reduce the acceleration change rate, and then proposed a power series-based inertia weight computing method. Therefore, the computing method of $\omega$ at the $d$-th iteration is shown in the formula (12).
The basic PSO is further discretized, and the variable \( I(Z_{u,d},\sigma,\omega) \) is defined to represent the state of the dimension \( \sigma + 1 \) \((\sigma < C)\) of the particle \( u \) at the \( d \)-th iteration, as shown in the formula (13). \( g(\star) \) is used to determine whether the state of \( \sigma + 1 \) has changed.

\[
I(Z_{u,d},\sigma,\omega) = g(Z_{u,d},\sigma,\omega) = \frac{1}{2}(1 + \exp(-\frac{y_{best}-y}{W_{max}}))
\]

(13)

The algorithm terminates when the MDPSO reaches 95% confidence \([24]\) or the number of iterations reaches the set value. The pseudo-code form of MDPSO is shown in the algorithm 1, where, "/**/" denotes the annotation.

**Algorithm 1.** Pseudo-code description of MDPSO

**Input:**
- \( l_{coalition\_structure\_Info} \): List of coalition structures; each coalition structure in the list is encoded in the form of the formula (4) and contains information such as utility \( y \), expected completion time \( ECT \), cost performance. This list forms the strategic space.
- \( l_{particle\_info} \): The dynamic information of the particle, including the current velocity \( V \), fitness \( y \), current position \( Z \), the historically optimal solution of the particle \( \nu_{best} \).
- Parameter setting: Learning factor \( c_1, c_2 \); Speed \( V_{max}, V_{min} \); Number of Iterations \( D \); Number of particles \( K \); Random number \( q_1, q_2 \); Inertia weight \( W_{max}, W_{min} \); Encoding length (number of members) \( C \).

**Output:**
- \( globe\_best \): Global optimal solution (optimal coalition structure)

**Begin**

1. for \( i=0 \) to \( l_{particle\_info}\_length \)
2. set \( l_{particle\_info}[i].V = \text{Random}(V_{min}, V_{max}); \) /*Speed of initialized particle*/
3. set \( l_{particle\_info}[i].Z = \text{Random}(l_{coalition\_structure\_Info}); \) /*Position of initialized particle*/
4. set \( l_{particle\_info}[i].y = \text{Fitness}(l_{particle\_info}[i].Z); \) /*Fitness of initialized particle*/
5. set \( l_{particle\_info}[i].v_{best} = l_{particle\_info}[i].y; \) /*Historically optimal solution of initialized particle*/

6. end

7. Update\(_{globe\_best, l_{particle\_info}}\); /*Update the global optimal solution by \( l_{particle\_info}\_length \)

8. for \( i=0 \) to \( D \)
9. for \( j=0 \) to \( K \)
10. set \( l_{particle\_info}[j].V = \text{W}^{*}l_{particle\_info}[j].V + c_1^{*}l_{particle\_info}[j].v_{best}^{*}l_{particle\_info}[j].y + c_2^{*}q_1^{*}(\text{globe\_best}^{*}l_{particle\_info}[j].y); \) /*Update speed*/
11. set \( s_{row} = (L+1)^{*}(L+1)\times(\exp(-l_{particle\_info}[j].V)); \) /*Update s\_row by the formula (16)*/
12. set \( l_{particle\_info}[i].Z = \text{Switch}(s_{row}); \) /*Update position by the formula (18)*/
13. set \( l_{particle\_info}[i].y = \text{Fitness}(l_{particle\_info}[i].Z); \) /*According to the current fitness of the particle*/
14. Update\(_{l_{particle\_info}[i].v_{best}, l_{particle\_info}[i].Z}; \) /*Update the historically optimal solution of the particle by \( l_{particle\_info}[i].Z\)*/
15. Update\(_{globe\_best, l_{particle\_info}}\); /*Update the global optimal solution by \( l_{particle\_info}\_length \)
16. end for
17. end for
18. return \( globe\_best \)

**End**

According to the Algorithm 1, the maximum number of executions of MDPSO is the product of the number of iterations and the number of particles. The number of particles and the number of iterations affect the search range and coverage of MDPSO, respectively. In other words, increasing the number of iterations can improve the quality of the optimal solution to some extent; increasing the number of particles can expand the search range, thereby reducing the possibility of “falling into the local optimal solution”. This also reflects the role of learning factors \( c_1, c_2 \).
The calculations of the algorithm come from the calculation of the fitness function. In one execution cycle, the fitness function will be calculated by three stages.

**Stage 1.** The member consuming the longest time in $M_j$ is taken for $f(M_j)$, and the number of calculations produced is $C$.

**Stage 2.** The utility of all coalitions constituting $M_j$ is accumulated for $v(M_j)$. According to the formula (6), the number of calculations of $v(M_j)$ is $(L + 1) \times (C + 1)$.

**Stage 3.** According to the formula (6) and (7), the number of calculations of $b(M_j)$ is $(L + 1)^2 \times (C + 1)$.

To sum up, MDPSO features the lowest computational complexity, and the maximum calculation is only

$$sum = K \times D \times ((L + 2) \times (C + 1) \times (L + 1) + C).$$

4. Experiments and Analysis

4.1 Experimental Preparation

**Experimental environment:** The subsystem of the edge computing system of an onshore oilfield was simulated. In the experimental environment, two high-performance servers were used to simulate the cloud computing center and data center, and a virtual machine was used to simulate the edge node (VM1-VM16). During the experiment, the Manage-Engine intelligent operation system was used to control the network bandwidth of each computing node. At the same time, a separate virtual machine (VMC) was established as a control center to make the scheduling strategy.

**Task information:** Taking the calculation of the heat resistance of polymer quasi-nanostructures as the engineering background, different equivalent polymers were selected to represent different tasks [22]. Table 1 shows the calculations, the planned completion time and the reward of the task.

**Environmental parameters:** The environmental parameters given in Table 1 are shown in Table 2, where the average expected completion time = the total task calculations / the total computing power of the experimental environment.

**Contrast algorithm:** HIDPSO [23] and Mobile Resource Awareness Scheduling Algorithm (MRA) [1] were selected as the contrast algorithm. The design principle and use scenarios of HIDPSO are similar to that of MDPSO, and HIDPSO was used to compare the performance of MDPSO. The MRA algorithm is a resource scheduling algorithm for the load balancing of the local mobile field, which is used to compare the local search capability of MDPSO.

| Table 1. Calculations, planned completion times, and reward for pre-designed tasks |
|-----------------------------------------------|-------|----------------|-------|
| Task ID | C(million of times) | Planned completion time (s) | Reward |
|--------|---------------------|-----------------------------|--------|
| T1     | 410                 | 115                         | 487    |
| T2     | 561                 | 105                         | 610    |
| T3     | 715                 | 100                         | 780    |
| T4     | 858                 | 120                         | 950    |
| T5     | 1073                | 110                         | 1195   |
| T6     | 1202                | 130                         | 1395   |
| T7     | 1367                | 125                         | 1520   |
| T8     | 1522                | 130                         | 1705   |

| Table 2. The experimental environment parameters analyzed and calculated |
|-----------------------------------------------|-------|--------|
| Parameter Interpretation | Parameter Values |
| Maximum number of tasks $L$                  | 8     |
| Maximum number of members $C$                | 16    |
| Number of possible coalition structures $W$  | $1.85 \times 10^{15}$ |
Total computing power of the experimental environment: 77.65
Total expenditure on experimental environment (per second): 80.25
Total task calculations (million of times): 7708
Planned completion time of tasks (maximum/minimum/average): 100/130/115.6
Total reward on completing all tasks: 8642
Average expected completion time: 99.26
Sum of computation complexity of fitness function: 2464

4.2 Performance Experiment of MDPSO

Under the conditions of different number of members, tasks, particles and iterations, such indexes as the mean run time \([24]\) (95% confidence), the total utility of the optimal coalition structure, and the expected completion time (ECT) were compared for the three algorithms. This is to verify the advantages of MDPSO in terms of convergence speed, global load balancing capability, expected completion time, fault tolerance, and relative stability.

During the experiment, the standard deviation (SD) will be used to measure the stability of the algorithm. Table 3 shows the basic parameter settings of the algorithm. The default value of the weight coefficient tends to coalition rationality, that is, to pursue the fast completion time and the high residual value \([25]\).

| Parameters | \(D\) | \(K\) | \(V_{max}\) | \(V_{min}\) | \(c_{1}\) | \(c_{2}\) | \(w_{max}\) | \(w_{min}\) | \(w_{r}\) | \(w_{s}\) |
|------------|------|------|-----------|-----------|-------|-------|-----------|-----------|-------|-------|
| Parameter values | [100,1000] | [5,50] | 7 | -7 | 1.9 | 1.9 | 3 | 0.5 | 0.1 | 0.8 | 0.1 |

4.2.1 Performance Experiment on Overall Load Balancing of Resources

Under the conditions of different number of members and tasks, the utility of the optimal coalition structure obtained, the expected completion time, and the mean run time of the optimization process were tested for the three algorithms, which is to observe the effect of the number of members and tasks on the algorithm’s ability to search the global optimal solution. The experimental results were averaged after removing obvious error points (local optimal solutions with deviation exceeding 10%). We set the maximum number of iterations \(D=500\) and the number of particles \(K=25\).

In the performance experiment with different number of members, the number of members is set as \(C \in [9,16]\). In order to prevent the occurrence of empty solutions (due to the large calculations of the task, any coalition structure cannot complete the task on time), four tasks with the smallest calculations (task number: T1-T4) were selected for the experiment. The experimental results are shown in Table 4. Members represent the member participating in the experiment.

In the performance experiment with different number of tasks, the number of members is set to 16 (that is, all members participate in the experiment). When the number of tasks is less than 4 \((L<4)\), the number of strategies is less than \(4.29 \times 10^9\), VMC can quickly obtain the optimal coalition structure through the enumeration method, so the minimum number of tasks participating in the experiment is set to 4. The experimental results are shown in Table 5. Tasks indicate the task number participating in the experiment.

By analysing the experimental results, we obtained the following conclusions.

1. The MRA algorithm is a local optimal search algorithm, so the total utility, the expected completion time (ECT) and other indexes of the optimal solution are obviously similar to or weaker than that of the other two algorithms, but the mean run time (MRT) has obvious advantages (reduced by about 60% compared with MDPSO), indicating that it has a strong dynamic adaptive ability; meanwhile, under the premise of ensuring the overall load balancing, the search speed of MDPSO is significantly higher than that of HIDPSO.
(2) At $C=9$ and $C=13$, the utility of the optimal coalition structure of the three algorithms is significantly improved; at the same time, the expected completion time of HIDPSO and MDPSO is significantly reduced. This is caused by the addition of members VM8 and VM4 with strong computing power, resulting in the change in the strategic space.

(3) When the number of members continues to increase, the expected completion time keeps unchanged in the MRA algorithm, but it changes significantly after adding VM3 (floating about 9%). This is because the addition of VM3 replaces the previous VM6 with strong computing power, so that VM6 can provide more resources to process other tasks with larger calculations, which avoids idle resources, and also indirectly indicates that MRA has good processing capacity for the load balancing of local resources.

Table 4. Performance test results with different number of members

| C   | Members | Utility          | ECT (s)          | MRT (ms)          |
|-----|---------|------------------|------------------|------------------|
|     |         | HIDPSO | MRA  | MDPSO | HIDPSO | MRA  | MDPSO | HIDPSO | MRA  | MDPSO |
| 8   | 9-16    | 223±2 | 222±6 | 245±1 | 116.4±0.2 | 118.6±0.5 | 115.2±0.7 | 430±2 | 163±1 | 267±3 |
| 9   | 8-16    | 348±3 | 336±8 | 368±2 | 109.7±0.2 | 118.2±0.3 | 108.1±0.3 | 431±3 | 165±2 | 266±2 |
| 10  | 7-16    | 357±2 | 344±6 | 369±1 | 109.5±0.3 | 117.9±0.1 | 108.5±0.5 | 431±2 | 157±2 | 267±4 |
| 11  | 6-16    | 362±2 | 344±7 | 370±2 | 109.7±0.2 | 117.6±0.9 | 108.6±0.5 | 432±3 | 161±3 | 266±2 |
| 12  | 5-16    | 351±3 | 344±7 | 367±3 | 108.9±0.6 | 117.3±1.1 | 107.9±0.8 | 431±2 | 162±3 | 267±2 |
| 13  | 4-16    | 401±3 | 397±6 | 428±2 | 101.4±0.7 | 117.5±0.9 | 101.7±0.7 | 432±4 | 163±2 | 267±1 |
| 14  | 3-16    | 419±5 | 415±9 | 430±2 | 101.7±0.4 | 108.3±1.2 | 101.8±0.2 | 430±1 | 163±3 | 266±3 |
| 15  | 2-16    | 423±4 | 428±7 | 428±2 | 102.1±0.3 | 108.9±1.1 | 101.4±0.5 | 430±4 | 164±1 | 265±2 |
| 16  | 1-16    | 426±2 | 424±8 | 429±1 | 102.3±0.5 | 108.7±0.9 | 101.6±0.3 | 431±1 | 159±1 | 266±3 |

Table 5. Performance test results for different number of tasks

| L  | Task | Utility          | ECT (s)          | MRT (ms)          |
|----|------|------------------|------------------|------------------|
|    |      | HIDPSO | MRA  | MDPSO | HIDPSO | MRA  | MDPSO | HIDPSO | MRA  | MDPSO |
| 4  | T1-4 | 380±2 | 378±9 | 422±1 | 117.5±1.3 | 118.6±1.7 | 115.6±1.7 | 430±2 | 159±8 | 265±3 |
| 5  | T1-5 | 416±2 | 414±7 | 457±3 | 115.9±2.0 | 116.4±0.8 | 113.7±1.3 | 432±1 | 160±4 | 266±2 |
| 6  | T1-6 | 451±3 | 436±9 | 487±2 | 115.4±1.5 | 115.3±1.4 | 113.2±2.2 | 431±3 | 162±4 | 264±4 |
| 7  | T1-7 | 490±1 | 474±8 | 515±3 | 113.4±1.6 | 114.2±0.9 | 111.9±1.4 | 432±2 | 161±3 | 265±1 |
| 8  | T1-8 | 517±3 | 486±9 | 548±2 | 111.3±1.3 | 113.7±0.8 | 110.7±0.7 | 430±1 | 160±1 | 266±3 |

4.2.2 Contrast Experiment on Optimization Ability of PSO

Under the conditions of different number of iterations and particles, the indexes such as the mean run time, the utility of the obtained optimal coalition structure, and the expected completion time were tested for HIDPSO and MDPSO, which is to analyze the influence of different number of iterations and particles on the optimization ability of the two algorithms. The experimental results were averaged after removing obvious error points (local optimal solutions with deviation exceeding 10%).

10 groups of experiments were conducted for different number of iterations, and the number of iterations increased from 100 to 1000 (a group of experiment was performed for each additional 100 iterations), and each group of experiment was performed for 5 times. Figure 1(a)-(d) show the effect of different number of iterations on the optimization performance of the two algorithms.

10 groups of experiments were performed for different number of particles, and the number of particles increased from 5 to 50 (a group of experiment was performed for each additional 5 particles), and each group of experiment was performed for 5 times. Figure 1(e)-(h) show the effect of different number of particles on the optimization performance of the two algorithms.

As a supplementary explanation, Table 6 shows the index values of the two algorithms in the
experiment. Among them, $\bar{D}$ and $\overline{ECT}$ indicate the number of iterations to obtain the optimal solution and the corresponding expected completion time of the optimal solution, and Error Rate indicates the frequency of falling into the local optimal solution.

Figure 1. Performance comparison of the two algorithms under different number of iterations and particles

Table 6. The index values of the two algorithms in experiments under different number of iterations and particles

| Test condition          | Index | HIDPSO   | MDPSO   |
|-------------------------|-------|----------|---------|
| Different iterations    | $K$   | 25       | 25      |
|                         | Maximum Utility | 518±2   | 550±1   |
|                         | $\overline{ECT}$ (s) | 111.2±1.1 | 110.9±1.4 |
| Different number of particles | $\bar{D}$ | 673±13  | 485±4   |
|                         | Maximum MRT (ms) | 687±3   | 559±4   |
|                         | Minimum MRT (ms) | 430±3   | 268±2   |
|                         | Max/Min SD       | 60/-59  | 10/-9   |
| Different number of particles | $D$   | 1000     | 1000    |
|                         | Maximum Utility  | 517±2   | 549±2   |
By analysing the experimental results under different number of iterations, the following conclusions were given.

As shown in Figure 1(a), the two algorithms converge and obtain the optimal solution after a certain number of iterations (485 times for MDPSO and 673 times for HIDPSO). The optimal solution quality and the convergence speed of MDPSO are better than that of HIDPSO. As shown in Figure 1(c), the mean run time of the two algorithms increases as the number of iterations increases. Because the algorithm terminates when the optimal solution is obtained, the mean run time of the two will stop growing after reaching a certain peak. As shown in Figure 1(d), the fluctuation range of HIDPSO is significantly higher than that of MDPSO, indicating that the algorithm stability of MDPSO is better than that of HIDPSO. Both of them quickly converge after approaching the optimal solution, because the algorithm terminates after reaching the optimal solution.

As shown in Figure 1(e), the two algorithms will quickly obtain the optimal solution when the number of particles is large enough, indicating that the larger the number of particles, the faster the optimal solution is updated. At the same time, it is further verified that the optimal solution quality of MDPSO is always slightly better than that of HIDPSO (not more than 6%). Figure 1(h) describes the probability that the two algorithms “fall into the local optimal solution” under different number of particles in the form of a histogram. Obviously, when the number of particles is small, the algorithm is more likely to fall into a local optimum, because the number of particles that can update the global optimal solution ($P_{g-old}$) is too small.

After further comparison and analysis of the experimental results, the following conclusions were drawn.

1) As shown in Figure 1(b) and Figure 1(f), the two algorithms will take the expected completion time of the coalition structure as the optimization objective, but the degree of optimization is not high (not more than 10%), because the preset weight coefficient tends to the total utility of the coalition structure ($w_c = 0.8$).

2) The utility of the optimal coalition structure obtained by the two algorithms is 518 and 550 respectively (floating less than 6%). This is because HIDPSO selects VM1 to participate in completing T2 and VM5 to participate in completing T1; and MDPSO selects VM1 to participate in completing T5 and VM5 to participate in completing T2. The fundamental reason is that the inertia weight of HIDPSO remains unchanged, thus missing the optimal solution.

5. Conclusion
We proposed an optimal coalition structure solution method to solve the resource scheduling problem of overall load balancing for edge computing in a large strategic space. A new expression of the coalition structure was constructed through this method, in addition, we established a multi-objective optimization problem model, adopted and improved the DPSO, thus realizing the fast search of the optimal coalition structure. The experimental verification results show that compared with similar algorithms, MDPSO has obvious advantages in global search capabilities, fault tolerance and stability, and thus is more suitable for solving the resource scheduling problem proposed in this paper.

Edge computing is an inevitable trend in the development of computing models, and the problem of resource optimization and task scheduling in different scenarios is also one of the focuses in related fields. When applying the method proposed in this paper to real scenarios, it is necessary to face the actual situations such as dynamic changes in resource bottlenecks and fluctuations in environmental computing power, which also need to be solved in the future. However, in the face of complex environmental factors, the method based on machine learning is worth trying.
References

[1] Pei Z, Piao S, Souidi M E H, et al., “Coalition Formation for Multi-agent Pursuit Based on Neural Network,” Journal of Intelligent & Robotic Systems 95.3-4 (2019) 887-899.

[2] Hoefer M, Vaz D, Wagner L, “Dynamics in matching and coalition formation games with structural constraints,” Artificial Intelligence 262.1 (2018) 222-247.

[3] Ueda S, Iwasaki A, Conitzer V, et al., “Coalition structure generation in cooperative games with compact representations,” Autonomous Agents and Multi-Agent Systems 32.4 (2018) 503-533.

[4] Zhan S, Niyato D, “A Coalition Formation Game for Remote Radio Head Cooperation in Cloud Radio Access Network,” IEEE Transactions on Vehicular Technology 66.2 (2018) 1723-1738.

[5] Dutta A, Dasgupta P, Baca J, et al., “Coalition Structure Environment Uncertainty Coalition Game Successor Node Good Node,” Distributed Autonomous Robotic Systems 104.1 (2016) 177-191.

[6] Xun F, Li D, “A new axiomatization of the Shapley-solidarity value for games with a coalition structure,” Operations Research Letters 46.2 (2018) 163-167.

[7] Shi W, Fellow, IEEE, et al., “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal 3.5 (2016) 637-646.

[8] Chen W, Wang D, Li K, “Multi-user Multi-task Computation Offloading in Green Mobile Edge Cloud Computing,” IEEE Transactions on Services Computing 12.5 (2019) 726-738.

[9] Alia M, Siarry P, Pant M, “An efficient Differential Evolution based algorithm for solving multi-objective optimization problems,” European Journal of Operational Research 217.2 (2012) 404-416.

[10] Félix Villafañez, Poza D, Adolfo López-Paredes, et al., “A generic heuristic for multi-project scheduling problems with global and local resource constraints (RCMPSP),” Soft Computing 23.10 (2019) 3465-3479.

[11] J.C. Yang, B. Jiang, Z.H. Lv, et al., “A task scheduling algorithm considering game theory designed for energy management in cloud computing,” Future Generation Computer Systems (2017) S0167739X17304673.

[12] S. Farzi, S. Yousefi, J. Bagherzadeh, et al., “Zone-based load balancing in two-tier heterogeneous cellular networks: a game theoretic approach,” Telecommunication Systems 70.1 (2019) 105-121.

[13] Y.P. Zhou, N.P. Hu, “Single machine scheduling problem with cost constraints based on customer driven and its non-cooperative game,” Control and Decision Conference IEEE 3 (2016) 2177-2180.

[14] G.H. Qiao, S.P. Leng, M. Zeng, et al., “Matching game approach for charging scheduling in vehicle-to-grid networks,” IEEE International Conference on Communications 5 (2017) 1-11.

[15] Wang Y, Sheng M, Wang X, et al., “Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling,” IEEE Transactions on Communications 64.10 (2016) 4268-4282.

[16] Jia M, Liang W, Xu Z, et al., “QoS-Aware Cloudlet Load Balancing in Wireless Metropolitan Area Networks,” IEEE Transactions on Cloud Computing (2018) 1-1.

[17] Zhang KJ, Fu Y, Hu YN, et al., “Scheduling Strategy for Computational-intensive Data Flow in Generalized Cluster Environments,” Applied Soft Computing 9.6 (2019) 1-24.

[18] Sarathambekai S, Kandaswamy U, “Task Scheduling in Distributed Systems Using Heap Intelligent Discrete Particle Swarm Optimization,” Computational Intelligence 33.4 (2017) 737-770.

[19] Chunlin L, Yanpei L, Youlong L, et al., “Collaborative content dissemination based on game theory in multimedia cloud,” Knowledge-Based Systems 15.124 (2017).
[20] Zhang KJ, Hu YN, Li CS, et al., “Scheduling Strategy of Compute-intensive Task-flow in Generalized Cluster,” Control and Decision 8.1 (2019) 1742-1753.

[21] Born R G, Kenevan J R, “Theoretical Performance-Based Cost-Effectiveness of Multicomputers,” The Computer Journal 35.1 (1992) 62-70.

[22] Rameshkumar K, Suresh R K, Mohanasundaram K M, “Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan,” Advances in Natural Computation 36.12 (2015) 572-581.

[23] Zhang H, Xie J, Hub Q, et al., “A hybrid DPSO with Levy flight for scheduling MIMO radar tasks,” Applied Soft Computing 71 (2018) 242-254.

[24] Anderson L M, Statistics With Confidence. “Confidence Intervals and Statistical Guidelines,” Journal of Clinical Pathology 12.88 (2000) 1439-1445.

[25] Bonnevay S, Kabachi N, Lamure M, “Agents-Based Simulation of Coalition Formation in Cooperative Games,” IEEE/WIC/ACM Joint International Conference on Intelligent Agent Technology, Compiegne (France), ACM (2005).