An Efficient Algorithm of Discrete Particle Swarm Optimization for Multi-Objective Task Assignment*

Nannan QIAO\textsuperscript{(a)}, Jiali YOU\textsuperscript{(b)}, Nonmembers, Yiqiang SHENG\textsuperscript{(c)}, Member, Jinlin WANG\textsuperscript{(d)}, and Haojiang DENG\textsuperscript{(e)}, Nonmembers

SUMMARY In this paper, a discrete particle swarm optimization method is proposed to solve the multi-objective task assignment problem in distributed environment. The objectives of optimization include the makespan for task execution and the budget caused by resource occupation. A two-stage approach is designed as follows. In the first stage, several artificial particles are added into the initialized swarm to guide the search direction. In the second stage, we redefine the operators of the discrete PSO to implement addition, subtraction and multiplication. Besides, a fuzzy-cost-based elite selection is used to improve the computational efficiency. Evaluation shows that the proposed algorithm achieves Pareto improvement in comparison to the state-of-the-art algorithms.

key words: task assignment, multi-objective, discrete particle swarm optimization, two-stage method

1. Introduction

Large granularity services such as multimedia effects and big data analysis emerges and develops rapidly in recent years. Flexible service architectures such as cloud computing\textsuperscript{(1)} and fog computing\textsuperscript{(2)} provides new opportunities for complex services by offering a number of virtual machines in a distribute way. The most common user demand of task assignment is to get a satisfactory time performance. Other performance such as lower budget for computation and transmission is also worthy of concerning.

In the past few years, Cloud computing\textsuperscript{(1)} has provided many opportunities for enterprises by offering a range of computing services. To guarantee real-time performance, other network-aware computing paradigms have been presented. In these paradigms, they utilize devices closed to users so that the reliable bandwidth and lower latency can be guaranteed. W. Zhu\textit{et al.} proposed a media-edge cloud (MEC) architecture\textsuperscript{(3)}, in which storage, central processing unit (CPU), Digital Object Identifier and graphics processing unit (GPU) clusters are presented at the edge to provide distributed parallel processing and QoS adaptation for various types of devices. Fog computing\textsuperscript{(2)} proposed by Cisco is viewed as a novel architecture built on the edge servers that provides the limited computing, storing, and networking services in a distributed way. It provides the logical intelligence to end users. The primary objective of fog computing is to ensure the low and predictable latency in the latency-sensitive applications such as media services.

The pay-as-you-go model becomes an efficient alternative to owning and managing private data centers for customers facing web applications and batch processing. Thus, the budget for resource occupation should also be taken into consideration in the task assignment problems.

Our research on task assignment is exactly based on the following environment.

1. All of the servers and users are located on the edge of the network, so the scheduler should be sensitive to the latency between a user to different servers.
2. The pay-as-you-go model is used to calculate the budget for task assignment. That is, the budget is positively related to the utilization of resource.

1.1 Related Work for Task Assignment

Many task scheduling algorithms based on best resource selection (BRS)\textsuperscript{(4)} have been proposed. Max-min and Min-min\textsuperscript{(5)} are two constructive methods in assigning independent tasks to processors in heterogeneous computing systems. In Max-Min, the scheduler always firstly assigns the task who has the maximal expect completion time (ect) to the instance which provides this \textit{ect}, then it updates the \textit{ect} and prepares for the next task. Min-Min is similar to Max-Min, and it always firstly assigns the task who has the minimal \textit{ect}. Another well-known constructive heuristics is Suffarge\textsuperscript{(6)}, which evaluates the priority for each task by calculating the difference between its minimal and second-minimal \textit{ect}. BRS-based methods are popular and well developed since they are free of parameters and easy to implement. However, if we consider more than one parameter which may be in conflict, it is convenient to use intelligent algorithms\textsuperscript{(7), (8)} rather than deterministic methods.

Evolutionary algorithms provide a more robust and efficient approach for real-world problems than traditional computation system. The most famous branch of evolutionary computation is genetic algorithm (GA), which has been...
proposed and employed in both optimization and machine learning. Solutions of a task assignment problem can be expressed as chromosomes in GA, and can share information with others by crossover and mutation operators. However, GA has high computation complexity for selecting target points and employing both crossover and mutation operators. PSO algorithm, which has gained much attention in a variety of fields, was firstly proposed by Kenny and Eberhard in 1995. This population-based algorithm comes from the behavior of bird flocks and fish schools. According to standard PSO, each particle adjusts the position according to the global best members and personal best members. However, as the particle positions are all real numbers, it can not be directly used for problems in discrete spaces such as scheduling and routing problems. So, in 1997, Kenny and Eberhard proposed a discrete binary particle swarm optimization approach [9] to produce binary coordinates of particles. Afterwards, many PSO-based algorithms have been developed based on parameters selection [10], position encoding and decoding [11–13], velocity strategy [14] and neighborhood structure [15].

1.2 Contributions of This Work

Although many swarm intelligence algorithms based on PSO have been developed, in this paper, we only focus on three aspects: 1) The way to encode the position of a particle, 2) The global best members selection rule, 3) The elite selection method in swarm evolution. Our main work is to minimize both makespan and budget simultaneously, where makespan is the total completion time of a batch of tasks, while budget denotes the price for computation and transmission resource. The contributions of this work are as follows:

1. Redefine the PSO operators and the meaning of learning factors. In this way, it is easier to get the discrete value of a new particle.
2. Design a new rule for selecting the global best members (gbest). In this method, we have two global guiders. The first one is chosen only according to makespan, while the second one is chosen according to budget. Evaluation shows that this rule is more effective than existing Pareto-front-based selection rule [16], [17].
3. Design a cost function based on fuzzy set and use elite selection according to cost value. The elite selection is effective in budget control, and get lower budget than single-objective approach [14] which only focus on price saving.

The rest of the paper is organized as follows. Section 2 describes the task assignment problem. In Sect. 3, we present our multi-objective discrete particle swarm optimization (MDPSO) algorithm. The evaluation and comparison are described in Sect. 4, and Sect. 5 draws the conclusion.

2. Problem Description

We model a workflow with \( N \) tasks as \( \{T_1, T_2, \ldots, T_N\} \). \( M \) servers \( \{S_1, S_2, \ldots, S_M\} \) located on the edge network support distributed processing. A feasible solution \( x \) indicates the machine assignment to tasks. For example, when \( N = 4 \) and \( M = 3 \), if \( x = [3, 2, 1, 3] \), the task-server mapping can be expressed as Fig. 1.

The symbols are defined as Table 1.

When a task \( T_i \) is assigned to \( S_j \), we use \( et_{i,j} \) to denote its expected completion time, and use \( b_{i,j} \) to denote its budget. The formulas for calculating \( et_{i,j} \) and \( b_{i,j} \) are as follows.

\[
\begin{align*}
\text{et}_{i,j} &= \text{ifs}_i / \text{mips}_j \\
\text{st}_{i,j} &= \max (\text{at}, \text{it}_{j}) + \text{ttr}(u, j, \text{ifs}_i) \\
\text{ect}_{i,j} &= \text{st}_{i,j} + \text{et}_{i,j} \\
b_{i,j} &= \text{bps}_j \ast \text{et}_{i,j} + \text{btr}(u, j, \text{ifs}_i)
\end{align*}
\]

Finally, applying these formulas, given tasks \( \{T_1, T_2, \ldots, T_N\} \), servers \( \{S_1, S_2, \ldots, S_M\} \) and the solution \( x = \{x_1, x_2, \ldots, x_N\} \), we calculate the makespan and budget following steps below.

1. Initial makespan(\( x \)) and budget(\( x \)) as 0;
2. For \( i = 1 \) to \( N \), repeat step 3-step 5;
3. Calculate \( et_{i,x_i} \) and \( b_{i,x_i} \) according to Eq. (1);
4. budget(\( x \)) = \( b_{i,x_i} \);
5. Update \( \text{it}_{x_i} \);
6. Mark the completion time of the last finished task (e.g., max(\( et_{i,x_i} \))), and set the makespan(\( x \)) as max(\( et_{i,x_i} \)) – at.

![Fig. 1](image-url)  

A feasible solution.

<table>
<thead>
<tr>
<th>Table 1 Problem symbols.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N ) the number of arrival tasks</td>
</tr>
<tr>
<td>( M ) the number of servers set of arrival tasks</td>
</tr>
<tr>
<td>( {T_1, T_2, \ldots, T_N} ) set of servers assignment for arrival tasks</td>
</tr>
<tr>
<td>( {S_1, S_2, \ldots, S_M} ) assignment for ( T_i ) the file size that any server ( S_j ) can process per second</td>
</tr>
<tr>
<td>( x_1, x_2, \ldots, x_N ) budget per second for ( S_j ) the input file size of ( T_i ) the budget for sending ( \mu )MB data from ( S_{\phi_1} ) to ( S_{\phi_2} )</td>
</tr>
<tr>
<td>mips ( j ) of ( T_i ) the output file size of ( T_i ) the time span for sending ( \mu )MB data from ( S_{\phi_1} ) to ( S_{\phi_2} )</td>
</tr>
<tr>
<td>bps ( j ) at the arriving time of the batch of tasks</td>
</tr>
<tr>
<td>if ( s_i ) ( u ) the machine ID of the user who requests ( T_i )</td>
</tr>
<tr>
<td>ofs ( i ) ( it_{j} ) the earliest idle time of ( S_j ) the execution time ( S_j ) needs to process ( T_i )</td>
</tr>
<tr>
<td>of ( s_i ) ( at ) the arriving time of the batch of tasks</td>
</tr>
<tr>
<td>btr(( \phi_1, \phi_2, \mu )) the time span for sending ( \mu )MB data from ( S_{\phi_1} ) to ( S_{\phi_2} )</td>
</tr>
</tbody>
</table>

The symbols are defined as Table 1.

When a task \( T_i \) is assigned to \( S_j \), we use \( et_{i,j} \) to denote its expected completion time, and use \( b_{i,j} \) to denote its budget. The formulas for calculating \( et_{i,j} \) and \( b_{i,j} \) are as follows.

\[
\begin{align*}
\text{et}_{i,j} &= \text{ifs}_i / \text{mips}_j \\
\text{st}_{i,j} &= \max (\text{at}, \text{it}_{j}) + \text{ttr}(u, j, \text{ifs}_i) \\
\text{ect}_{i,j} &= \text{st}_{i,j} + \text{et}_{i,j} \\
b_{i,j} &= \text{bps}_j \ast \text{et}_{i,j} + \text{btr}(u, j, \text{ifs}_i)
\end{align*}
\]
The price for resource are set based on the following rule.

1. All of the servers and users are located in clusters on the edge network, and the available bandwidth between clusters is much lower than that within a cluster;
2. The expense for transmitting data between different clusters is positively related to the data size, and transmitting data within a cluster in free in our problem;
3. Servers with great computing power is always expensive because CPUs with high MIPS or large capacity memory deserve high price.

Our objective of the problem is to find a task-server mapping, assuring that both makespan and budget are minimized.

### 3. MDPSO Algorithm

#### 3.1 QoS Function

Many fitness functions for multi-objective problem have been proposed in previous work, the most common fitness is the weighted sum of all objectives. In our problem, we use a cost function to define the performance of a particle. A particle who brings shorter makespan and lower budget will get a low cost value. The cost function is defined as:

\[
C(x, \alpha) = \begin{cases} 
1 - \frac{f_1(x) + (1-\alpha)f_2(x)}{\max\{f_1(x), f_2(x)\}} & \text{if } \text{budget}(x) \in A \\
1 & \text{otherwise}
\end{cases}
\]

(2)

where \(x\) is a feasible solution, and \(\alpha\) is any number in \([0, 1]\).

Two fitness functions \(f_1(x)\) and \(f_2(x)\) are defined independently in Eqs. (3) and (4) where \(m_{base}\) and \(b_{base}\) are empirical values. \(f_1(x)\) is a fitness function in terms of \(x\)’s makespan, and \(f_2(x)\) only considers the \(x\)’s budget.

\(A\) is a fuzzy set and its membership function is shown as Eq. (5). That is to say, a particle \(x\) whose budget is more than a system-defined value \(b^*\) may have the risk to get the cost value \(C(x) = 1\). When its budget is more than the upper limit \(b^*\), it will certainly cause \(C(x) = 1\).

\[
f_1(x) = 1 - \text{makespan}(x)/m_{base}
\]

(3)

\[
f_2(x) = 1 - \text{budget}(x)/b_{base}
\]

(4)

\[
m_A(b) = \begin{cases} 
1 - (b - b^*)/(b^* - b^*) & b < b^* \\
0 & b \geq b^*
\end{cases}
\]

where \(b^*\) is any number in \([0, 1]\).

Then the objective of the task assignment problem is depicted as:

\[
\text{Minimize} : C = \{C(x, a_1), C(x, a_2), \ldots, C(x, a_n)\}
\]

(6)

In our task assignment problem, we simplify the cost function by setting \(n = 2\), and \(a_1 = 1\), \(a_2 = 0\). Then the objective function can be stated as:

\[
\text{Minimize} : C = \{C(x, a_1), C(x, a_2)\}
\]

(7)

where

\[
C(x, a_1) = \begin{cases} 
1 - \frac{f_1(x)}{\max\{f_1(x), f_2(x)\}} & \text{if } \text{budget}(x) \in A \\
1 & \text{otherwise}
\end{cases}
\]

(8)

\[
C(x, a_2) = \begin{cases} 
1 - \frac{f_2(x)}{\max\{f_1(x), f_2(x)\}} & \text{if } \text{budget}(x) \in A \\
1 & \text{otherwise}
\end{cases}
\]

(9)

#### 3.2 Global Best Members Selection Rule

Although makespan and budget are in conflict with each other in terms of the computing power of servers, they have some consistency. In bandwidth-intensive situation, transmitting too many tasks across clusters may cause a lot delay time, and it is also not profitable for expense. So, the two objectives of the problem are not completely conflict with each other. To some degree, they are consistent because reducing the data between clusters is benefit to both makespan and budget. So we select global guiders \(g_{best}\) and \(g_{best_2}\) based on the single-objective fitness functions, and try to get the consistent information of them. In each iteration, we sort all particles according to the value of \(f_1(x)\) and put the top-\(L_1\) solution into \(\text{Archive}_1\), then put the top-\(L_2\) particles according to \(f_2(x)\) into \(\text{Archive}_2\). In \(t\)th iteration, each particle \(x_t\) chooses global guiders \(g_{best}^{1}\) and \(g_{best}^{2}\) randomly from \(\text{Archive}_1\) and \(\text{Archive}_2\).

#### 3.3 Particle Encoding

Since each particle means a task-machine assignment, it is unnecessary for a particle to keep the previous path. Actually, a particle can "fly" toward any guider. So, in this problem, the personal information of each particle is neglected. In each iteration, \(x_t^{i}\) only updates its velocity and position according to \(g_{best}^{1}\) and \(g_{best}^{2}\). The addition, subtraction and multiplication operators for updating position are designed. The process of position updating is depicted as follows.

\[
\tilde{x}_t^{i+1} = \tilde{c}_r^1 \odot \left( g_{best}^{1} \odot \tilde{x}_t^{i} \right) + \tilde{c}_r^2 \odot \left( g_{best}^{2} \odot \tilde{x}_t^{i} \right)
\]

(10)

\[
\tilde{x}_t^{i+1} = \tilde{x}_t^{i} \oplus \tilde{v}_t^{i+1}
\]

(11)

where \(\tilde{c}_r^1\) and \(\tilde{c}_r^2\) are two N-dimension vectors, each bit of which compromises 0 or 1. Each bit of a particle denotes the ID of the target server. The manners in which the operators work is as Figs. 2, 3 and 4, and the meaning is explained as follows.

The \(\odot\) operator: We use this operator to find the difference between a particle \(x\) and \(g_{best}\) and then retain the unique information of \(g_{best}\). For example, for \(P[p_1, p_2, \ldots, p_N]\) and \(Q[q_1, q_2, \ldots, q_N]\), if \(p_i\) is unequal to \(q_i\), we set the \(i\)th bit of \(P \odot Q\) equal to \(p_i\), otherwise, the \(i\)th bit of \(P \odot Q\) is 0.

The \(\oplus\) operator: This operator is used to randomly reserve part of information of \(g_{best}\). For example, \(P\) is a binary sequence with \(N\) bits. When \(p_i\) is 1, we set the \(i\)th bit of \(P \oplus Q\) as \(q_i\), and else, we set it as 0.
Fig. 2 The \(\ominus\) operator.

\[
\begin{array}{ccccccc}
\rho & 14 & 4 & 13 & 9 & 5 & 7 \\
\uparrow & \downarrow & \uparrow & \downarrow & \uparrow & \downarrow & \uparrow \\
\sigma & 13 & 5 & 13 & 23 & 5 & 7 \\
\end{array}
\]

\[
P \odot Q = 14 \ 4 \ 0 \ 9 \ 0 \ 0
\]

Fig. 3 The \(\otimes\) operator.

\[
\begin{array}{ccccccc}
\rho & 1 & 0 & 1 & 0 & 0 & 1 \\
\uparrow & \downarrow & \uparrow & \downarrow & \uparrow & \downarrow & \uparrow \\
\sigma & 13 & 5 & 13 & 23 & 5 & 7 \\
\end{array}
\]

\[
P \odot Q = 13 \ 0 \ 13 \ 0 \ 0 \ 7
\]

Fig. 4 The \(\oplus\) operator.

\[
\begin{array}{ccccccc}
\rho & 3 & 0 & 1 & 7 & 0 & 9 \\
\uparrow & \downarrow & \uparrow & \downarrow & \uparrow & \downarrow & \uparrow \\
\sigma & 3 & 5 & 0 & 23 & 5 & 7 \\
\end{array}
\]

\[
P \oplus Q = 3 \ 5 \ 1 \ 7 \ 5 \ 9
\]

3.4 The 2-Stage MDPSO Algorithm

3.4.1 Initialization Stage with Elite Members

We firstly initialize the swarm with random solutions. Then, inspired by the work of K.B. Lee and J.H. Kim [18], elite members are added into the initial swarm.

In order to provide elite members, we add three artificial particles into the swarm based on Max-Min, Min-Min and Suffrage Method. These methods are all good at load balance, thus they have shorter makespan than random solutions. Besides, in order to provide solutions with low budget, we add three particles based on Localization idea, that is, the tasks are randomly assigned to the same cluster with the user. This idea prevents data transmission across different clusters, so the budget for bandwidth can be largely saved.

3.4.2 Evolution Stage

We adopt a fuzzy set based selection in the proposed algorithm in order to simplify computation. That is to say, in each iteration, we knock out the particle \(x\) if \(C(x) = 1\). The evolution terminates when the number of iterations is more than \(\text{Iterations}\) or the number of remain particles is less than a default number. The detail of the proposed MDPSO algorithm is as Algorithm 1.

4. Evaluation

In this section, evaluations were carried out to compare approaches. All the approaches were coded in Matlab 7.1 and run on a PC on Windows with Intel i5-4570 3.20 GHz CPU and 3.43GB available memory.

We build 14 VMs distributed in 3 clusters for task execution. The price for data transmission between clusters is set in Table 2, which is in similar level with Amazon EC2.
The computing power shown in Table 3 denotes the file size that a VM can process per second, while the price means the budget per second of a running machine. The data transmission speed between clusters is set as 2.5Mbps, while the speed within a cluster is 12.0Mbps. Problems with small (15 tasks), medium (50 tasks), large (100 tasks) size were implemented as benchmark problems, and the file size of tasks is uniformly distributed in 32–64MB.

The proposed MDPSO method is compared with existing Pareto approaches and single-objective approaches. For a fair comparison, the local search and the operations priority were not employed in all approaches. In order to get tradeoff of makespan and budget, we set price-oriented and time-oriented instances. In price-oriented instances ($I_1$, $I_3$ and $I_5$), in order to allow more particles evolving in a budget saving way, we let $g_{best1}$ play a more important role than $g_{best1}$ in guiding the swarm, so we set $\rho_1 = 0.3$ and $\rho_2 = 0.5$, thus $g_{best1}$ has more probability to affect the particle position. Besides, the budget constraints are strictly limited, that is, the values of $\xi_1$ and $\xi_2$ in these instances are set lower. In time-oriented instances ($I_2$, $I_4$ and $I_6$), we extended the budget constraints and increased the value of $\rho_1$. Since the parameters of comparative approaches in time-oriented instances are as same as those in price-oriented instances, their results in $I_2$, $I_4$ and $I_6$ are omitted and considered as the same with $I_1$, $I_3$ and $I_5$ respectively.

For all instances, the termination criterion is to reach the maximal number of generations. Each algorithm runs for 20 times independently. The makespan, budget, weighted summation of objectives of each approaches are calculated for comparison, and the formula of weighted summation is as Eq. (12), where $b_{base}$ and $m_{base}$ are mean values of makespan and budget of initial members, and the weights of the objectives are set as $W_1 = W_2 = 0.5$. The budget constraint in membership function (Eq. (5)) is set as follows, where $Avr(budget_{init})$ is the average budget of all members in initialization. The probability that each bit of learning factors $c_{1}$ and $c_{2}$ gets one is expressed as $\rho_1$ and $\rho_2$, and the number of elements in $Archive_1$ and $Archive_2$ is set as $L$.

$$W_1 \times \frac{\text{Makespan}(x)}{m_{base}} + W_2 \times \frac{\text{Budget}(x)}{b_{base}}$$  \hspace{1cm} (12)

$$b^* = \xi_1 \times Avr(budget_{init})$$  \hspace{1cm} (13)

$$b^* = \xi_2 \times Avr(budget_{init})$$  \hspace{1cm} (14)

### 4.1 Comparison with Existing Pareto Approaches

In this subsection, the proposed MDPSO is compared with existing Pareto approaches [16], [17]. F. Ramezani et al. proposed a Pareto approach [16] by randomly selecting the global best member from the top-10 particles in Pareto archive (PSO+RS method). G. Moslehi and M. Mahnam designed a method based on the sigma value of Pareto optimal members [17] (PSO+Sigma method). These literatures provided the details about the machine selection rule according to a particle position, which makes it possible for us to implement them on our task-assignment problem. As the learning factors and weights were not reported in the PSO+RS method, we use parameters as suggested in [17]. As the PSO+RS method needs at least ten members in the archive, in order to get enough Pareto optimal solutions, we set the swarm size as 50, 100, and 200 in different instances. All the parameters are listed in Table 4. In each instance, the solutions of all the three approaches are integrated to find the overall Pareto front. The average results of each instances are shown as Table 5. The makespan and budget of 20 independent executions are shown as Figs. 5, 6, 7, 8, 9 and 10.

As can be seen from Table 5, in price-oriented instances ($I_1$, $I_3$ and $I_5$), although we decreased the budget by average 8.3–17.8% than PSO+RS approach and 11.8–12.8% than PSO+Sigma approach, the makespan is more than them because of the strict limit of budget. In time-oriented instances ($I_2$, $I_4$ and $I_6$), both makespan and budget were decreased in contrast to Pareto approaches.
Table 5  Comparison with existing Pareto approaches.

<table>
<thead>
<tr>
<th>Instance</th>
<th>MDPSO</th>
<th>PSO+RS [16]</th>
<th>PSO+Sigma [17]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2195</td>
<td>2062</td>
<td>2062</td>
</tr>
<tr>
<td>2</td>
<td>1890</td>
<td>5525</td>
<td>9879</td>
</tr>
<tr>
<td>3</td>
<td>5957</td>
<td>115</td>
<td>115</td>
</tr>
<tr>
<td>4</td>
<td>5312</td>
<td>966</td>
<td>990</td>
</tr>
<tr>
<td>5</td>
<td>11131</td>
<td>115</td>
<td>455</td>
</tr>
<tr>
<td>6</td>
<td>9674</td>
<td>115</td>
<td>990</td>
</tr>
</tbody>
</table>

The makespan of MDPSO are 1890, 5312 and 9674 in time-oriented instances, which is 3.3%–9.8% lower than PSO+RS approach and 2.1%–8.3% lower than PSO+Sigma approach. The budget of MDPSO is approximate to that of the PSO+Sigma method in the small problem instance $I_2$, but in medium and large problem instances, the budget of MDPSO is 5.9% and 4.3% lower than PSO+Sigma method. Besides, MDPSO decreased budget by average 1.2%–6.5% in contrast to PSO+RS approach. Evaluation shows that the proposed $g_{best}$ selection rule is more effective in information sharing.

4.2 Comparison with Single-Objective Approach

Xingquan Zuo et al. proposed a SLPSO-based scheduling
Table 6 Parameters of SLPSO approach and proposed MDPSO approach.

<table>
<thead>
<tr>
<th>Size</th>
<th>SLPSO [14]</th>
<th>MDPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of tasks</td>
<td>Swarm Size</td>
</tr>
<tr>
<td>$I_1$</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>$I_2$</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>$I_3$</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>$I_4$</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>$I_5$</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>$I_6$</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 7 Comparison with SLPSO approach.

<table>
<thead>
<tr>
<th>Method</th>
<th>MDPSO</th>
<th>SLPSO [14]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Makespan</td>
<td>6588</td>
<td>6460</td>
</tr>
<tr>
<td>Budget</td>
<td>67</td>
<td>75</td>
</tr>
<tr>
<td>Weighted Summation</td>
<td>0.73</td>
<td>0.77</td>
</tr>
<tr>
<td>Runtime</td>
<td>0.218</td>
<td>0.222</td>
</tr>
</tbody>
</table>

Fig. 10 Comparison with existing Pareto approaches in $I_6$.

Fig. 11 Comparison with SLPSO approach in $I_1$.

Fig. 12 Comparison with SLPSO approach in $I_2$.

We implemented the four velocity updating strategies and the selection rule of the strategies in SLPSO approach. However, because we had some different settings with SLPSO, we made modification as follows.

1. SLPSO [14] used the number of available CPU and the size of memory to simulate the situation of local resource shortage, and limited the number of tasks as-
signed to local VMs. Because we didn’t set CPUs and memory, we modified the SLPSO method using MaxLoad of local cluster. That is, MaxLoad denotes the number of tasks that can be supported by local cluster, and it was set in the range of \([0, N]\), where \(N\) is the total number of tasks.

2. As SLPSO considered the resource of external clouds as infinite, so it is possible to control the completion time no more than predefined deadline. However, in our settings each VM has limited computing power, and the total completion time is positively related to the number of tasks, so we neglected the time deadline in this simulation.

3. As MDPSO needs more computation time because of multiplication operation, we increased the iterations of SLPSO to make sure its runtime no less than MDPSO. The iterations and average runtime are listed in Table 6.

We can see that, in price-oriented instances \((I_1, I_3, \text{and } I_5)\), the estimated makespan of SLPSO in small, medium and large instances are \(10708, 29080\) and \(65218\) respectively, and the makespan of MDPSO approach are \(6588, 19799\) and \(47025\), which were \(38.5\%\), \(32.0\%\) and \(27.9\%\) lower than SLPSO. The budget was decreased by \(19.5\%\), \(19.6\%\), and \(23.4\%\), from \(83, 340\) and \(635\) of SLPSO to \(67, 274\) and \(487\) of MDPSO.

In time-oriented instances \((I_2, I_4, \text{and } I_6)\), because we adjusted the weight of two objective and extended the budget constraint, the makespan of MDPSO is lower than price-oriented instances, with the cost of more budget. However, in \(I_2, I_4, \text{and } I_6\), MDPSO also decreased the makespan by \(39.7\%, 42.0\%\) and \(35.8\%\), and decreased budget by \(9.8\%\) and \(14.8\%\). As SLPSO only focused on task priority and assigned tasks with high priority to local VMs, without regard to task execution time, MDPSO could largely decrease makespan since it is a multi-objective approach. In addition, the elite selection based on fuzzy set in swarm evolution stage was more effective in limiting budget than SLPSO which controlled budget by adjusting task priority.

5. Conclusion

In this paper, we proposed a discrete particle swarm optimization algorithm (MDPSO) to solve the multi-objective task assignment problem in distributed environment. A two-stage approach was designed for minimizing both makespan and budget for arrival tasks. A new method for position encoding, a gbest selection rule, and a fuzzy-cost-based elite
selection were employed. The efficiency of the MDPSO approach was compared with results of existing Pareto approaches [16], [17] and single-objective approach [14]. The evaluation results indicated that, in comparison to Pareto approaches [16], [17], MDPSO achieved the multi-objective optimization in most runs of time-oriented instances, but in price-oriented instances, as the budget was strictly limited, MDPSO achieved minimal budget and weighted summation at the cost of more makespan. In comparison to single-objective SLPSO approach which focus on price saving, MDPSO completely dominated the SLPSO approach in all of the instances. The evaluation results indicated that MDPSO approach is competitive compared to the state-of-art algorithms. The proposed global best members selection rule is more effective in information sharing because it provides guiders with high quality. Besides, the fuzzy-cost-based elite selection also has the ability to control the overall budget by knocking out invalid solutions.

Acknowledgments

The authors would like to thank the anonymous reviewers for the constructive comments that greatly improved this article.

References


Nannan Qiao received her bachelor’s degree from Beijing University of Posts and Telecommunications, Beijing, China in 2013. She is going to get a master’s degree in National Network New Media Engineering Research Center, Chinese Academy of Sciences, Beijing, China. Her current research interests are in optimization algorithms, media service and big data.

Jiali You is an associate professor of the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences. She received her B.S. in Computer Science from Communication University of China in 2003, and Ph.D. in Signal and Information Processing from the Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS) in 2008. From July 2008, she joined the National Network New Media Engineering Research Center, IOA, CAS. Her research interests include distributed network, cloud computing, and future network.
Yiqiang Sheng received his Master’s degree from Nankai University, Tianjin, China, in 2003 and his Doctoral degree from Tokyo Institute of Technology, Tokyo, Japan, in 2014. He worked at Sanyo Electric Co., Ltd. in Japan from 2003 to 2008 and at Nokia Corporation in Tokyo, Japan, from 2008 to 2010. He did research at Osaka University, Osaka, Japan, from 2010 to 2011 and at Tokyo Institute of Technology, Tokyo, Japan, from 2011 to 2014. He is currently with National Network New Media Engineering Research Center, Chinese Academy of Sciences, Beijing, China, as academic researcher and associate professor. His current research interests are in smart systems, optimization algorithms, machine learning, big data and network theory with applications.

Jinlin Wang is a professor of the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences (CAS). He received his B.S. from University of Science and Technology of China in 1986, and Master’s degree from the Institute of Acoustics, CAS in 1988. From 1988 he joined Institute of Acoustics, CAS. Now he is the chairman of National Network New Media Engineering Research Center. His research interests include digital signal processing, broadband information network, and multimedia communication.

Haojiang Deng is a professor of the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences (CAS). He received his B.S. from Wuhan University in 1993, Master’s degree from China Academy of Space Technology in 1998, and Ph.D. from the Institute of Semiconductors, CAS in 2001. From September 2001, he joined the National Network New Media Engineering Research Center, Institute of Acoustics, CAS. His research interests include pattern recognition, video information processing, and multimedia communication.