INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2015; **28**:1053–1067
Published online 3 February 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.2743

A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment

Fengyu Guo[1], Long Yu[2,*,†], Shengwei Tian[3] and Jiong Yu[3]

[1]*Graduate School of Information Science and Engineering, Xinjiang University, Urumqi, China*
[2]*Network Center, Xinjiang University, Urumqi, China*
[3]*School of Software, Xinjiang University, Urumqi, China*

SUMMARY

Cloud computing is the key and frontier field of the current domestic and international computer technology, workflow task scheduling plays an important part of cloud computing, which is a policy that maps tasks to appropriate resources to execute. Effective task scheduling is essential for obtaining high performance in cloud environment. In this paper, we present a workflow task scheduling algorithm based on the resources' fuzzy clustering named FCBWTS. The major objective of scheduling is to minimize makespan of the precedence constrained applications, which can be modeled as a directed acyclic graph. In FCBWTS, the resource characteristics of cloud computing are considered, a group of characteristics, which describe the synthetic performance of processing units in the resource system, are defined in this paper. With these characteristics and the execution time influence of the ready task in the critical path, processing unit network is pretreated by fuzzy clustering method in order to realize the reasonable partition of processor network. Therefore, it largely reduces the cost in deciding which processor to execute the current task. Comparison on performance evaluation using both the case data in the recent literature and randomly generated directed acyclic graphs shows that this algorithm has outperformed the HEFT, DLS algorithms both in makespan and scheduling time consumed. Copyright © 2014 John Wiley & Sons, Ltd.

Received 2 January 2013; Revised 19 December 2013; Accepted 24 December 2013

KEY WORDS: cloud computing; workflow task scheduling; fuzzy clustering; resources division

1. INTRODUCTION

With the rapid development of the Internet, the distribution range of the Internet is expanding gradually, the network resources are still increasing strongly. Cloud computing is a new proposed calculation model that can provide dynamic and scalable virtual resources through the Internet service to users on demand, and also it is further development of distributed computing, parallel computing, and grid computing [1–5].

Through accomplishing the overall sharing of various resources and being combined with the cloud resource characteristics—heterogeneity, dynamic, flexibility, and so on— cloud computing can deal with the amount of datasets to complete all the computing needs of users. Nevertheless, it puts forward higher requirements on the techniques of task scheduling [6] and resource allocation. Workflow technology [7, 8] provides an effective solution to resolve these problems in cloud computing, and workflow scheduling becomes a research focus. Most of the applications of workflow scheduling research are based on the directed acyclic graph, for instances, Modified Critical Path [9], Dynamic Critical Path (DCP) [10]. These algorithms are mainly applied to multiprocessor systems in homogenous environment, and they ignore the heterogeneity characteristics of resources in cloud computing. There are many typical workflow task scheduling algorithms, which are in heterogeneous

*Correspondence to: Long Yu, Network Center, Xinjiang University, Urumqi, China.
†E-mail: yulong_xju@126.com

environment, such as Heterogeneous Earliest Finish Time (HEFT) [11] and Dynamic Level Scheduling (DLS) [12]. These methods are based on the quantitative characteristics of the task and do not consider the service-oriented resources. And it is difficult to describe exactly the task demands for resources, meanwhile, resource attributes cannot be described accurately. Fuzzy theory provides various effective means to solve the uncertain problems in the real world. So, in this paper, we use fuzzy clustering to divide the resources and propose a fuzzy clustering based workflow task scheduling algorithm to improve the efficiency of task scheduling.

The rest of the paper is organized as follows: we present in Section 2 a few related works. Then, in Section 3, we establish the scheduling models and describe the process of fuzzy clustering for the processors. Section 4 introduces the proposed scheduling algorithm. In Section 5, we make a comparison study of our algorithm with others, and all these algorithms are based on randomly generated directed acyclic graphs (DAG). The summary of the presented research and planned future work are given in Section 6.

2. RELATED WORK

The major idea of cloud computing is to split the massive computing processing tasks into many smaller sub-tasks through the network and hand over the sub-tasks to huge systems composed by multiple servers, then pass the results back to users after search, computation, and analysis. The network that provides resources is called 'cloud'. Because of the huge number of computing tasks in cloud computing environment, the task scheduling and resource allocation are the difficult problems in determining the efficiency of cloud computing.

At present, cloud computing research is at an early stage, task scheduling problem has been one of the key research contents of the computer science. Scheduling algorithms in cloud environment get the attentions of numerous scholars. According to different emphases in the scheduling, Jia Yu *et al*. [13] divided the task scheduling algorithm into two kinds of heuristic task scheduling algorithms: task level and workflow level. The task-level scheduling algorithm determined the scheduling strategy in light of the local information of a single task or series of independent tasks, whereas workflow-level scheduling algorithm needed to consider all the information of entire workflow tasks to adjust the scheduling strategy. Opportunistic Load Balancing algorithm and Sufferage algorithm [14, 15] are the task-level heuristic task scheduling algorithms that are widely used. Most of the aforementioned algorithms select all resources directly, they can be applied to the distributed environment with a small number of resources, but the superior performance will be vanished in cloud environments with rich resources. Daoud *et al*. [16] proposed Longest DCP (LDCP) algorithm to deal with the task scheduling problem with dependent tasks relations in heterogeneous environments. LDCP algorithm defined a DAG processor (DAGP) for each resource, scheduled the critical path in DAGP, but the proposed hypothesizes were not fully established. Lan *et al*. [17] redefined critical task and idle time slice, they proposed a critical task scheduling algorithm Heterogeneous critical task (HCT) in heterogeneous environment, HCT algorithm copied the critical task to the resources with idle time slice to advance the start time of tasks.

These research algorithms only concern about how to reduce the execution time of the workflow tasks and have simple hypothesis, for example, the same time of task execution, the limitation of the number of processing units, and the lack of analysis of resource characteristics. However, with the development of cloud computing and the expansion of related resources, the appropriate selection of processing resources is an important factor to determine task scheduling.

Clustering is an unsupervised classification method, the clustering process is classifying the physical or abstract objects according to the similarity between the objects. Clustering is an effective way to solve the problem of resource classification; the existing clustering methods can be divided into hierarchical-based, density-based, and web-based methods [18]. General clustering partition is processing each object strictly, whereas in the real environment, most objects have no clear property division. Because the fuzzy set theory [19–21] was put forward by Zadeh, researchers apply the fuzzy theory to clustering and get many achievements in specific areas. So, we use the

DOI: 10.1002/dac

fuzzy theory to analyze resource performance, preprocess the resources, and achieve the integration and classification of a large number of resources, then we improve the HEFT list sort and complete the task scheduling according to their priority.

3. PROBLEM DESCRIPTIONS

3.1. Cloud computing

Cloud computing is hinting at a future in which we would not compute on local computers but on centralized facilities operated by third-party compute and storage utilities [5]. Computing pioneer John McCarthy predicted that 'computation may someday be organized as a public utility', and went on to speculate how this might occur [3, 4]. There is little consensus on how to define the cloud computing, many computing researchers and practitioners have attempted to define clouds in various ways.

Buyya *et al*. [22] thought that cloud computing was a new and promising paradigm delivering services as computing utilities. Clouds were designed to provide services to external users, providers needed to be compensated for sharing their resources and capabilities. Wang L.Z *et al*. [23] defined the science cloud computing system, they pointed out that cloud computing could not only provide the hardware service HaaS (Hardware as a service), software service SaaS (software as a service) and data resource service DaaS (Data as a service), but also could provide the platform service PaaS (platform as a service) to the user. So the users would submit their own on-demand to the computing platform about the hardware configuration, software installation, data access requirements, *and so on*. With the increasing demand for process automation in the cloud, there is a requirement to create a collaborative workflow infrastructure as part of sophisticated problem solving processes [24].

Cloud workflow applications often consist of a several tasks, and each task is implemented by several substitute cloud services. Traditional workflow management systems should be to adapted to this new paradigm in order to leverage the benefits of cloud service. A scheduling schema in cloud computing workflow system is shown in Figure 1.

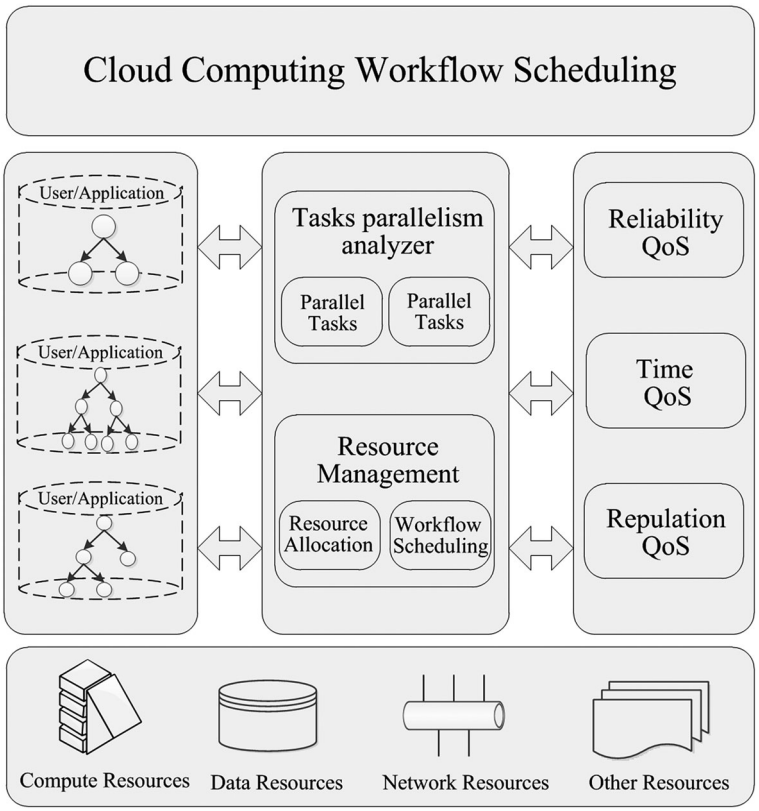


Figure 1. Cloud workflow scheduling schema.

DOI: 10.1002/dac

3.2. Task model

Cloud computing is the manifestation of parallel system, the workflow can reflect the characteristics of the tasks in cloud environment, a directed acyclic graph is a common way of descriptions. In the paper, a workflow is described as a tuple $G=(T,E)$, T is a set of tasks $T=\{t_1,t_2,\ldots,t_n\}$, $n=|T|$ is the number of the tasks. $W(t_i)$ represents the computation cost of the task $t_i \in T$. $E=\{e_{ij}|e_{ij}=(t_i,t_j), e_{ij}\in T\times T\}$ is a set of e_{ij} edges between the tasks, each e_{ij} represents the precedence constraint such that task t_i should complete its execution before task t_j starts. $C(t_i,t_j)$ was assigned to edge e_{ij} represents the communication cost between tasks t_i and t_j, if t_i, t_j are not in a same processor, the value will be zero if both t_i, t_j are mapped in the same processor.

In a given workflow task graph, $PRED(t_i)=\{t_j|t_j\in T, e_{ji}\in E\}$ is the set of immediate predecessors of t_i, and $SUCC(t_i)=\{t_j|t_j\in T, e_{ij}\in E\}$ is the set of immediate successors of t_i. In $G=(T,E)$, a task without any predecessors is called an entry task (t_{entry}) and a task without any successors is called an exit task (t_{exit}). If there is more than one entry (exit) task, they are connected to a zero-cost pseudo entry (exit) task with zero-cost edge, which does not affect the performance of the schedule. An example of DAG is shown in Figure 2, their execution costs are on the nodes and edge weights are written on the edges.

Before introducing FCBWTS algorithm, it is necessary to define the *EST* and *EFT* attributes, which are derived from a given partial schedule. $EST(t_i, p_j)$ and $EFT(t_i, p_j)$ are the earliest execution start time and the earliest execution finish time of the task t_i on the processor p_j, respectively. After a task t_k is scheduled on a processor p_j, the earliest start time of t_k on processor p_j is equal to the actual start time $AST(t_k)$, and the earliest finish time of t_k on the processor p_j is equal to the actual finish time $AFT(t_k)$. After all, tasks in a graph are scheduled, the overall completion time will be the actual finish time of the exit task t_{exit}. If there are multiple exit tasks, the schedule length (which is also called makespan) is defined as the following:

$$makespan = \max\{AFT(t_{exit})\} \tag{1}$$

The objective of the workflow task scheduling is to determine the assignment of tasks to processors such that its schedule length is minimized.

In order to describe the proposed workflow task model clearly, we make several definitions as the following:

Definition 1 *b-level:* The *bottom level* (*b-level*) of arbitrary task t_i is the length of the longest path from t_i to an exit task (including the right of the task t_i), denoted as $b\text{-}level(t_i)$.

Definition 2 *t-level:* The *top level* (*t-level*) of arbitrary task t_i is the length of the longest path from an entry task to t_i (excluding the weight of t_i).

Definition 3 Critical path: The critical path of a diagram is the longest path from the entry node to the exit node, including both nonzero communication edge cost and task weights in that path.

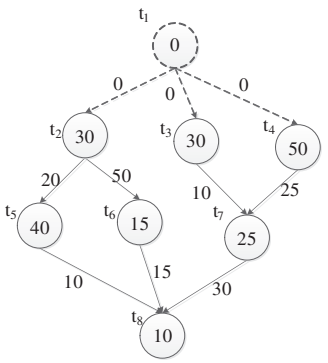


Figure 2. An example of task directed acyclic graph.

DOI: 10.1002/dac

The tasks on the critical path determine the schedule length of the workflow tasks. In a homogeneous environment, many algorithms rank the tasks by *b-level* or *t-level* in descending order, and then get ready task to schedule, these methods embody the priority of the task in critical path. We not only consider the critical path, but also the computation costs and communication costs in the heterogeneous environment, we improve the prior order of tasks. We add the mid-value of the computation costs (M_{comp}) and the mid-value of the communication costs (M_{com}) in processing unit network to measure the priority level of tasks, which will increase the accuracy of the algorithm. We use $priority(t_i)$ to indicate the priority of the task.

$$priority(t_i) = \frac{W(t_i)}{M_{comp}} + \max_{t_j \in SUCC(t_i)} \left(\frac{C(t_i, t_j)}{M_{com}} + priority(t_j) \right) \tag{2}$$

However, we redefine that $AST^*(t_i, p_j)$ is the actual start time of task t_i on processor p_j, $AFT^*(t_i, p_j)$ is the actual finish time of task t_i on processor p_j.

$$AFT^*(t_i, p_j) = AST^*(t_i, p_j) + W(t_i)/W(p_j) \tag{3}$$

When $t_j \in PRED(t_i)$ and the tasks t_i, t_j perform on the processing units p_x, p_y respectively, the arrived time of the tasks t_i, t_j on the processing units p_x, p_y is $AT(t_i, t_j, p_y)$, which is defined by the following:

$$AT(t_i, t_j, p_y) = AFT^*(t_i, p_x) + C(t_i, t_j)/C(p_x, p_y) \tag{4}$$

$W(p_j)$ and $C(p_x, p_y)$ can be seen in 3.2. $EST^*(t_j, p_y)$ is the earliest execution start time of task t_j on the processer p_j, t_k is the latest execution task on p_j.

$$EST^*(t_j, p_y) = \max \left\{ AFT^*(t_k, p_y), \max_{t_j \in PRED(t_i)} (AT(t_i, t_j, p_y)) \right\} \tag{5}$$

3.3. Resource model

Different from the traditional homogeneous environment, this paper assumes that the topological structure of the resource model is in a heterogeneous environment, its heterogeneity is reflected in two aspects: (1) the different processing capacity between processors and (2) the different transmission speed of different links. So, the resource model is represented by a weighted nondirective graph $G_P = (P, E_P)$, where $P = \{p_1, p_2, \ldots, p_n\}$ is a set of processing units, $n = |P|$ is the number of the processing units, $E_P = \{e_{ij} | e_{ij} = (p_i, p_j), e_{ij} \in P \times P\}$ is the set of edges. In the proposed model, it is also assumed that $W(p_i)$ is the computation in unit time of a processing unit $p_i \in P$ and $C(p_i, p_j)$ is the communication in unit time of the link $(p_i, p_j) \in E_P$; shown in Figure 3.

In the processing unit network, each unit can be computed and communicated simultaneously, but every link only transmits a message in each moment. If a message is transmitted through several links, their communication time is the sum of communication time on all links that it transmits.

3.4. Fuzzy clustering

The computation and communication ability of resources allocated by the task will affect the completion of its subsequent task, therefore, distinguishing the performance of the resources is conductive to choose the appropriate resources for the workflow task scheduling. But it is difficult

DOI: 10.1002/dac

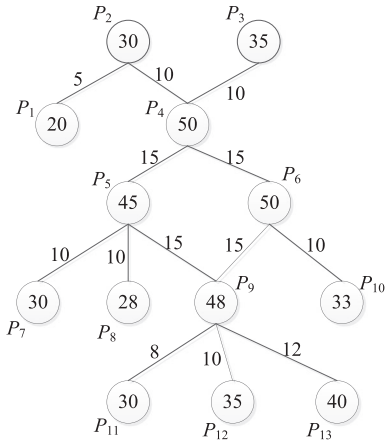


Figure 3. An example of resource network graph.

to describe the attributes of resources accurately, and there are no attributes to distinguish the processing units strictly. Fuzzy clustering is an effective method to divide resources.

3.4.1. Resource characteristic. According to the characteristics of the processing unit network, we firstly define several characteristics as follows:

(1) Processing capacity: the computation cost in unit time of the processing unit in resource network, it is the weight of processing unit.
(2) Average communication capacity: the mean value of the communication costs of the links that is connected with the processing unit, the capacity is the average weight value of the edges connected this processing unit.
(3) Maximum communication rate: the link that has the fastest transmission speed is connected with this processing unit.
(4) Network location: the variance of the distance from a processing unit to the center processing unit, it shows the position of the processing unit in the network. The higher variance indicates that this processing unit is located in the edge of the network. On the contrary, it is in the center of the network.
(5) The number of links: the number of the edges connected this processing unit.

3.4.2. Fuzzy clustering analysis. The fuzzy clustering of resources consists of three phases, and the specific processes are shown in Figure 4.

(1) Data standardization

We use the vector $Q_i = (q_{i1}, q_{i2}, \ldots, q_{im})$ to present the attributes' set of the ith processing unit, where m is the number of the attributes, we set $m = 5$, $Q_i = (q_{i1}, q_{i2}, q_{i3}, q_{i4}, q_{i5})$ represents the characteristics of the resource as defined earlier. We use the mean value of the characteristics' attributes to standardize the original data, eliminate the dimensional affection, and obtain data q'_{ik}.

$$q'_{ik} = \frac{q_{ik} - \overline{q}_k}{s_k} (i = 1, 2, \ldots, N; k = 1, 2 \ldots, m) \tag{6}$$

Where $\overline{q}_k = \frac{1}{N} \sum_{i=1}^{N} q_{ik}$ is the kth mean of the original data characteristics, $s_k = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (q_{ik} - \overline{q}_k)}$ is the standard deviation of the original data. Because the acquired data is not entirely in the interval [0, 1], we use formula (7) to process it:

DOI: 10.1002/dac

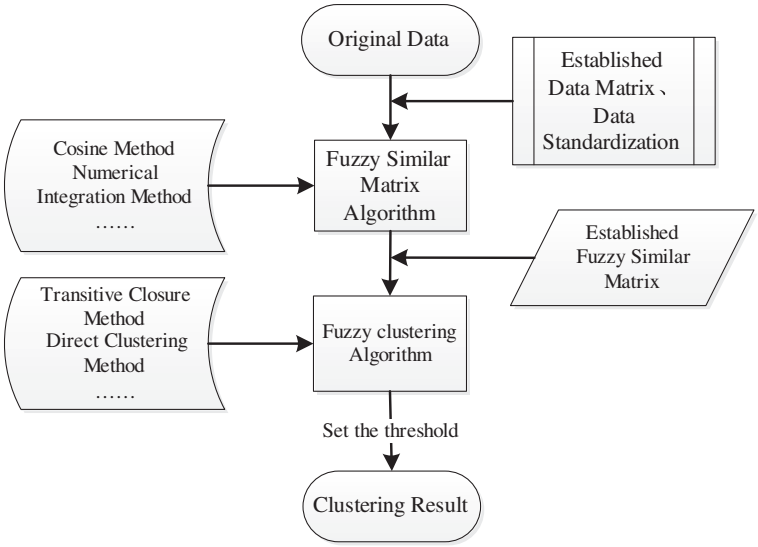


Figure 4. Fuzzy clustering process.

$$q''_{ik} = \frac{q'_{ik} - \min\left(q'_{ik}\right)}{\max\left(q'_{ik}\right) - \min\left(q'_{ik}\right)} \tag{7}$$

(2) Establishment of fuzzy similar matrix

We calculate the similarity between the processing units, as referred r_{ij} in formula (8).

$$r_{ij} = \begin{cases} 1 & i = j \\ [0,1] & i \neq j \end{cases} \tag{8}$$

Then, we can use the exponential function method of similar coefficient to get the fuzzy similarity and establish fuzzy similar matrix by the formula (9).

$$r_{ij} = \frac{1}{m}\sum_{k=1}^{m} e^{-\frac{3\left(q''_{ik} - q''_{jk}\right)}{4s^2{}_k}} \tag{9}$$

(3) Fuzzy clustering

On the basis of fuzzy similar matrix, we are calculating $\widetilde{R}$ by the formula (10) until r'_{ij} remains no change, so we can obtain the fuzzy equivalent matrix R^*.

$$r'_{ij} = r_{ij} = \bigvee_{k=1}^{N}\left(r_{ik} \wedge r_{jk}\right) = \max_{1 \le k \le N}\left(r_{ik} \wedge r_{jk}\right) \tag{10}$$

In fuzzy clustering analysis, we can get the different classifications by the different thresholds, thus, forming a kind of dynamic clustering figure, which is more comprehensive in the understanding of computer resources. But when we are clustering based on the resource characteristics, we can determine a threshold according to the specific characteristics. Similarity in the same threshold of resources is considered to be the same performance nodes. It is a

DOI: 10.1002/dac

kind of very good method to determine the threshold using statistical method, and it can adapt to the general computing tasks.

According to setting the different threshold $\alpha(0 \le \alpha \le 1)$, we can obtain the different clusters. The more similarity the threshold is, the higher the degree of the clusters similarity. The more α is closer to 0, the less similarity in clusters. Finally, we get the logic partitioning by the closure matrix R^* and threshold and get the clustering results.

4. PROPOSED ALGORITHM

The objective of this algorithm is to shorten the completion time of every task in the workflow, whereas the start time and the finish time are the two important factors of affecting the completion time. If the task is allocated to the different processing units, the execution implementation is also different. The performance of resource clusters generated by fuzzy clustering is used as a reference in our algorithm, we schedule each task on its suitable processing unit, which minimizes the finish time of task. The scheduling process of the proposed algorithm is shown in Figure 5.

4.1. Algorithm description

The FCBWTS algorithm is an application scheduling algorithm for the heterogeneous environment, which has three major phases: (1) a resource clustering phase for dividing the processors; (2) a task prioritizing phase for computing the priorities of all tasks; and (3) a processor allocation phase for scheduling the tasks on the suitable processor.

(1) Resource clustering phase

After extracting the information of resource network, we can get five characteristic information of processing unit and divide the processing units by fuzzy clustering model in quarter 3.2. Table I shows the characteristic of the original data according to the Figure 2.

We standardize the original data Q according to the formulae (6), (7) and get the fuzzy similar matrix $\widetilde{R}$ without the influence of dimension.

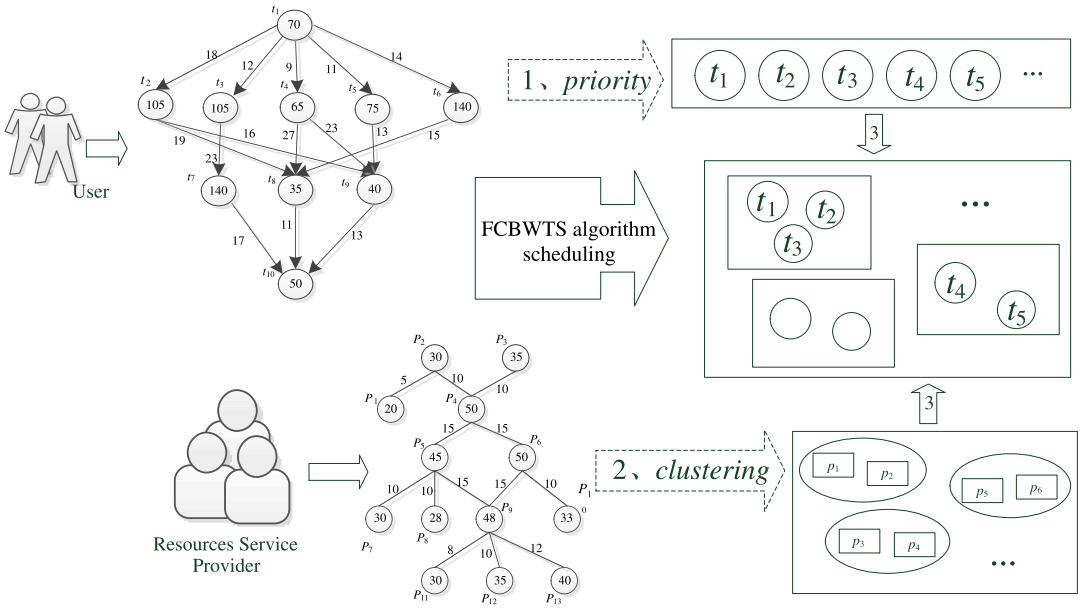


Figure 5. Algorithm for scheduling model.

DOI: 10.1002/dac

Table I. Resources' characteristics of the original data.

Feature					
Processing unit	q_1	q_2	q_3	q_4	q_5
p_1	20.00	0.05	0.05	6.00	1.00
p_2	30.00	0.08	0.05	9.00	2.00
p_3	35.00	0.10	0.10	10.00	2.00
p_4	50.00	0.13	0.10	14.00	4.00
p_5	45.00	0.13	0.10	13.00	4.00
p_6	50.00	0.13	0.10	14.00	3.00
p_7	30.00	0.10	0.10	9.00	1.00
p_8	28.00	0.10	0.10	8.00	1.00
p_9	48.00	0.11	0.03	13.00	5.00
p_{10}	33.00	0.10	0.10	9.00	1.00
p_{11}	30.00	0.08	0.08	8.00	1.00
p_{12}	35.00	0.10	0.10	10.00	1.00
p_{13}	40.00	0.12	0.12	11.00	1.00

$$\widetilde{R} = \begin{Bmatrix} 1.00 & 0.79 & 0.59 & 0.54 & 0.54 & 0.56 & 0.60 & 0.60 & 0.52 & 0.60 & 0.60 & 0.59 & 0.59 \\ 0.79 & 1.00 & 0.60 & 0.57 & 0.57 & 0.59 & 0.60 & 0.60 & 0.56 & 0.60 & 0.80 & 0.59 & 0.59 \\ 0.59 & 0.60 & 1.00 & 0.78 & 0.78 & 0.79 & 0.99 & 0.99 & 0.56 & 0.99 & 0.59 & 0.99 & 0.60 \\ 0.54 & 0.57 & 0.78 & 1.00 & 0.99 & 0.99 & 0.75 & 0.75 & 0.60 & 0.75 & 0.55 & 0.76 & 0.56 \\ 0.54 & 0.58 & 0.78 & 0.99 & 1.00 & 0.99 & 0.76 & 0.75 & 0.60 & 0.76 & 0.55 & 0.76 & 0.56 \\ 0.56 & 0.59 & 0.79 & 0.99 & 0.99 & 1.00 & 0.77 & 0.77 & 0.58 & 0.77 & 0.57 & 0.78 & 0.58 \\ 0.59 & 0.60 & 0.99 & 0.75 & 0.76 & 0.77 & 1.00 & 1.00 & 0.53 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.59 & 0.60 & 0.99 & 0.75 & 0.75 & 0.77 & 0.99 & 1.00 & 0.53 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.52 & 0.56 & 0.56 & 0.60 & 0.60 & 0.58 & 0.53 & 0.53 & 1.00 & 0.53 & 0.53 & 0.54 & 0.53 \\ 0.60 & 0.60 & 0.99 & 0.75 & 0.76 & 0.77 & 1.00 & 1.00 & 0.53 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.60 & 0.80 & 0.59 & 0.55 & 0.55 & 0.57 & 0.60 & 0.60 & 0.53 & 0.60 & 1.00 & 0.60 & 0.60 \\ 0.60 & 0.60 & 0.99 & 0.78 & 0.76 & 0.78 & 1.00 & 1.00 & 0.54 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.60 & 0.60 & 0.60 & 0.56 & 0.56 & 0.58 & 0.60 & 0.60 & 0.54 & 0.60 & 0.60 & 0.60 & 1.00 \end{Bmatrix}$$

Then, we obtain the fuzzy equivalent matrix R^* through the transitive closure method.

$$R^* = \begin{Bmatrix} 1.00 & 0.79 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.59 & 0.60 & 0.79 & 0.60 & 0.60 \\ 0.79 & 1.00 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.59 & 0.60 & 0.80 & 0.60 & 0.60 \\ 0.60 & 0.60 & 1.00 & 0.79 & 0.79 & 0.79 & 0.99 & 0.99 & 0.59 & 0.99 & 0.60 & 0.99 & 0.60 \\ 0.60 & 0.60 & 0.79 & 1.00 & 1.00 & 0.99 & 0.79 & 0.79 & 0.59 & 0.79 & 0.60 & 0.79 & 0.60 \\ 0.60 & 0.60 & 0.79 & 1.00 & 1.00 & 0.99 & 0.79 & 0.79 & 0.59 & 0.79 & 0.60 & 0.79 & 0.60 \\ 0.60 & 0.60 & 0.79 & 0.99 & 0.99 & 1.00 & 0.79 & 0.79 & 0.59 & 0.79 & 0.60 & 0.79 & 0.60 \\ 0.60 & 0.60 & 0.99 & 0.79 & 0.79 & 0.79 & 1.00 & 1.00 & 0.59 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.60 & 0.60 & 0.99 & 0.79 & 0.79 & 0.79 & 1.00 & 1.00 & 0.59 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.59 & 0.59 & 0.59 & 0.59 & 0.59 & 0.59 & 0.59 & 0.59 & 1.00 & 0.59 & 0.59 & 0.59 & 0.59 \\ 0.60 & 0.60 & 0.99 & 0.79 & 0.79 & 0.79 & 1.00 & 1.00 & 0.59 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.79 & 0.79 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.59 & 0.60 & 1.00 & 0.60 & 0.60 \\ 0.60 & 0.60 & 0.99 & 0.79 & 0.79 & 0.79 & 1.00 & 1.00 & 0.59 & 1.00 & 0.60 & 1.00 & 0.60 \\ 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.60 & 0.59 & 0.60 & 0.60 & 0.60 & 1.00 \end{Bmatrix}$$

DOI: 10.1002/dac

The objective of clustering is to divide the data to clusters, so we set the threshold $\alpha = 0.80$ (the higher the value is, the closer the performance in the clusters is). We obtain the dividing result:{{p_1}, {p_2}, {p_3,p_7, p_8, p_{10}, p_{12}}, {p_4, p_5, p_6}, {p_9}, {p_{11}}, {p_{13}}}. The clusters are ranked in descending order by the comprehensive properties :{p_4 , p_5 , p_6}, {p_9}, {p_{13}}, {p_3 , p_7 , p_8 , p_{10} , p_{12}}, {p_2}, {p_{11}}, {p_1}.

(2) Task prioritizing phase

Figure 6 shows a workflow task instance. We calculate the priority value for each task of the workflow and establish the ready node list of sets (*readylist*).

In this paper, we take into account the computing power of the link and the transmission capacity of the processing unit, so we determine the task priority values according to the attribute *priority* and record the task node on the critical path. The length of critical path is the sum of the computation costs of the tasks on the path and intertask communication costs along the path. If the attribute *priority* value of a task is greater, its priority is greater. If there is more than one critical path, we can select the task node on the path with the maximum traffic. When a task is on the top of the ready set list, we choose it to schedule. Because some tasks are released in every task allocation process, the ready set list will be updated after each task assignment. We calculate the attribute priority value of each workflow task in Figure 5, as shown in Table II.

(3) Processor allocation phase

The goal of the proposed algorithm is to shorten the execution makespan of the workflow task scheduling, therefore, we try to shorten the completion time of each task scheduling.

We divide all of the processing units into the optional set and the backup set (the overall performance of optional set is better than the backup set), calculate the maximum, minimum completion time of the workflow task t_i in the optional processing unit, and obtain the difference value between maximum completion time and minimum completion time. If the difference value is greater, the influence on task scheduling that assigned to the different processing units is greater. So, the task that has the greater difference value can get the higher priority and be scheduled in the processing unit to have the minimum completion time. For the difference value of smaller tasks, the completion times of the tasks are similar in the processing units, the scheduling in which process unit affects completion time less.

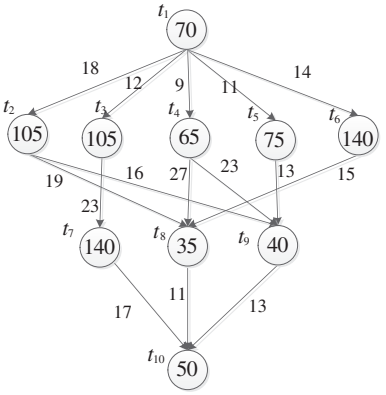


Figure 6. Workflow tasks graph.

Table II. The task priority of Figure 5.

Node	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Priority	15.63	8.47	12.43	8.09	7.31	9.03	7.13	3.53	3.87	1.43

DOI: 10.1002/dac

The task with the highest priority is scheduled on the processing unit that can complete the task with the minimum completion time (we consider the optional set first). When there is no vacancy in the optional set, we choose the processing unit of the cluster with the highest comprehensive performance in backup set and calculate the completion time of current task on this processing unit. If the completion time of this processing unit in backup set is smaller than the smallest completion time of the processing unit in optional set, we put the current task schedule on the one in backup set. If there are two processing units that have the same completion time, we can choose the processing unit whose network location is nearer to the center unit. Because of the consideration of communication cost, we place the tasks on the critical path of the workflow to the same processing unit as far as possible.

4.2. Algorithm analysis

HEFT algorithm only considers the global information when sorting the tasks, but does not take into account the communication overhead impact of the current task on the successor task execution. HEFT uses average processing time of task on the processing unit to replace the computation cost of specific task allocation, so it cannot determine the impact of the former assigned task to its successor and it is not easy to obtain the optimum dispatching sequence for HEFT. DLS is similar to the proposed algorithm, it gets the matching dynamic level between preparatory task and idle resource by computation. DLS also considers the heterogeneity of resource environment, but the quantization process of resources in our paper can express resource characteristics more accurately.

We use the fuzzy clustering method to divide the resources in the pretreatment process before scheduling, task scheduling will choose the processing unit with good comprehensive performance to allocate task instead of detecting the entire processing unit each time. The core codes of the proposed algorithm are described in Table III.

Table III. Key code of the FCBWTS algorithm.

```
1.  Set the computation costs of processing units and communication costs of edges with mid-value.
2.  Compute priority for all tasks by traversing graph upward, starting from the exit task.
3.  Sort the workflow tasks in a scheduling list by descending order of priority(t_i) values called
    readylist.
4.  if (process ∈ firstProcess&&!isempty(firstProcess))
5.     for each t_i ∈ readylist     //computing priority(t_i);
6.        if (priority(t_i)>maxValue)
7.           maxValue = priority(t_i);
8.           task = t_i;
9.        end if
10.    end for each
11.    makespan = AFT*(task , process);
12. else
13.    if (AFT*(task , process)<=makespan)
14.       min(computing(task , process));     // put task in the minimize process
15.    else
16.       if (AFT*(task , process')<AFT*(task , process))
17.          makespan = AFT*(task , process');
18.    end if
19. end if
```


DOI: 10.1002/dac

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1. Experimental environment

We chose MATLAB (The MathWorks Company, United States) for simulation experiments and used HEFT algorithm, DLS algorithm as the benchmark experiments, and compared the two algorithms with FCBWTS algorithm in order to obtain a relatively objective evaluation. In our study, we designed a random graph generator to ensure the accuracy of the simulation, the generator was depended on several input parameters given by the users for the workflow task graph and resource graph.

The random generator requires the following input parameters:

(1) n: number of tasks in the graph.
(2) m: of processing units in the graph.
(3) [min(*nodeWeight*), max(*nodeWeight*)]: the computation cost is randomly generated in it. min (*nodeWeight*) is the minimum weight of graph node, the max(*nodeWeight*) is the maximum weight of graph node.
(4) [min(*edgeWeight*), max(*edgeWeight*)]: the communication cost (the weight of an edge) is randomly generated in it.
(5) *maxOutDegree* : The maximum out degree of the nodes, the out degree of each node is a random integer in [0, *maxOutDegree*] and the out degree of the exit node is 0.
(6) *CCR*: Communication to computation ratio. It is the ratio of the average communication cost to the average computation cost.

The difference between the workflow task graph and the resource network graph is that workflow task graph must be DAG, so we use the parameter flag to distinguish the purpose of generated graph.

5.2. Performance Metrics

In order to evaluate the performance of the proposed algorithm, we adopt common performance comparison metrics based on the makespan: schedule length ratio (SLR), speedup.

Schedule length ratio is the main measure of a scheduling algorithm on a graph. It is defined by

$$SLR = \frac{makespan}{\sum_{t_i \in CP_{\min}} \min_{p_j \in P} \{w_{i,j}\}} \tag{11}$$

The denominator is the summation of the minimum computation costs of tasks on the $CP_{\min}$. The SLR of a graph cannot be less than one. The algorithm that has the lowest SLR is the best algorithm with respect to performance.

Speedup for the algorithm is computed by dividing the sequential execution time by the parallel execution time.

$$speedup = \frac{\min_{p_j \in P} \left\{ \sum_{t_i \in T} w_{i,j} \right\}}{makespan} \tag{12}$$

The sequential execution time is computed by assigning all tasks to a single processing unit that minimizes the computation costs.

5.3. Experimental results analysis

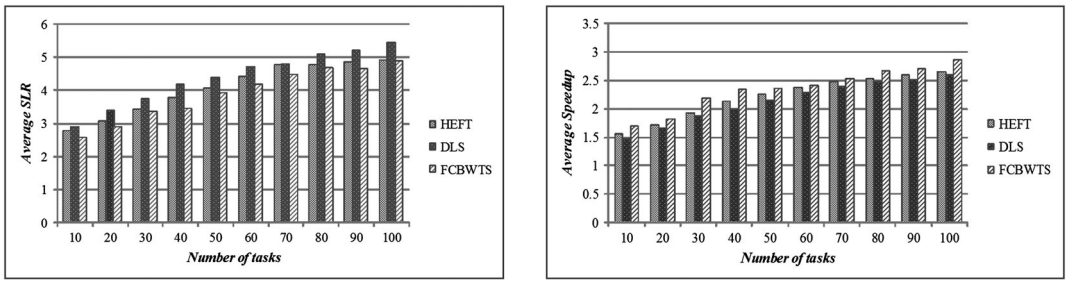
In this paper, we compare HEFT algorithm, DLS algorithm with the FCBWTS algorithm and obtain the following experiments.

In experiment 1, we set $n \in \{10,20,30,40,50,60,70,80,90,100\}$, $m = 10$, $CCR = 2.0$, the range of task computation cost and communication cost is [50,200], processing unit computation in unit time

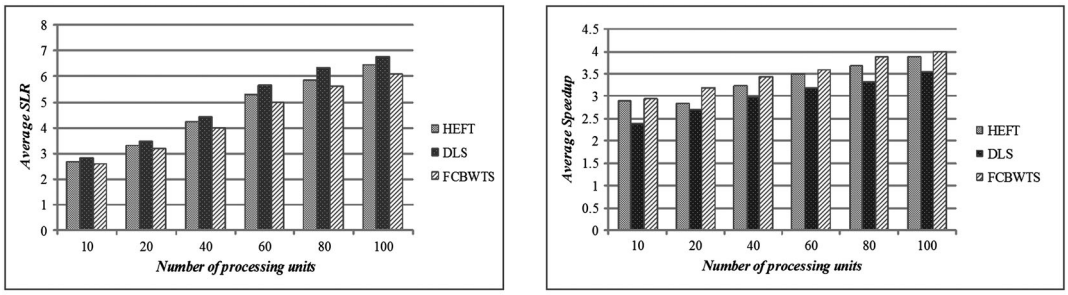
DOI: 10.1002/dac

is in the range of [10,150], and the range of processing unit communication in unit time is [10,100], $maxOutDegree = 4$. We choose the average value random experiments to measure the performances, and Figure 7(a) shows the obtained comparative results of SLR and speedup. Through the experimental contrast analysis, the average SLR value of FCBWTS is better than the HEFT algorithm and DLS algorithm by 4.8%and 12.5% respectively. On average, the speedup achieved is greater than the HEFT algorithm and the DLS algorithm by 6.3%, 10.3% respectively. This is mainly because we put forward a new prioritization way with the resource capacities, then this way could narrow the resource allocation of time and improve the scheduling the execution.

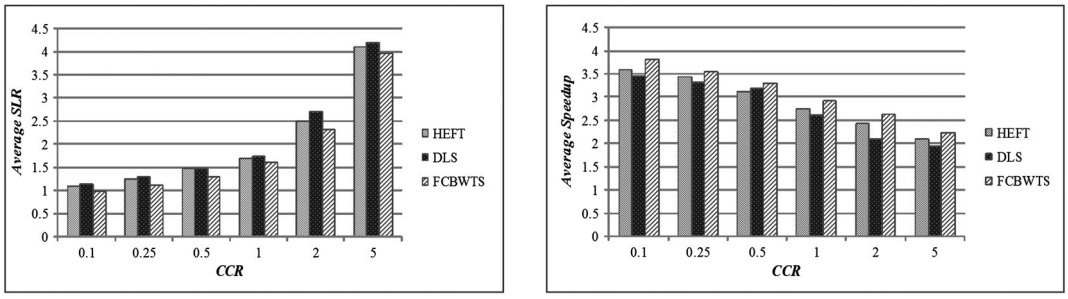
In experiment 2, we set $n = 50$, $m \in \{20,40,60,80,100\}$, and $CCR = 2.0$, the range of task computation cost and communication cost is [50,200], processing unit computation in unit time is in the range of [10,150] and the range of processing unit communication in unit time is [10,100], $maxOutDegree = 4$. The experimental result is shown in Figure 7(b). Based on these experiments, the disparity of these algorithms is less when there are not many resources. With the increase of the resources, the parallel of task execution is enhanced. FCBWTS algorithm is better than others. The analysis of results shows that the average SLR value of FCBWTS is better than the HEFT algorithm by 5.1%, the DLS algorithm by 11.1%, and the acquired speedup is greater than the HEFT algorithm by 4.8%, the DLS algorithm by 16.2%.



(a) Performance for each algorithm with changing tasks' number



(b) Performance for each algorithm with changing processing units' number



(c) Performance for each algorithm with changing *CCR*

Figure 7. Diagrams of experiment for performance comparison. (a) Performance for each algorithm with changing tasks' number. (b) Performance for each algorithm with changing processing untis' number. (c) Performance for each algorithm with changing *CCR*.

DOI: 10.1002/dac

In experiment 3, we set $n = 50$, $m = 10$, and $CCR \in \{0.1,0.25,0.5,1.0,2.0,5.0\}$, the range of task computation cost and communication cost is [50,200], processing unit computation in unit time is in the range of [10,150] and the range of processing unit communication in unit time is [10,100], $maxOutDegree = 4$. Fifty DAGs in the experiment are randomly generated, the SLR and speedup is shown in Figure 7(c). The analysis of results shows that the average SLR value of FCBWTS is better than the HEFT algorithm by 8.6%, the DLS algorithm by 12.8%, and the speedup acquired is greater than the HEFT algorithm by 5.9%, the DLS algorithm by 12.1%.

6. CONCLUSIONS

In this paper, we summarized the traditional workflow task scheduling algorithm in cloud computing environment, and found some factors that affect the scheduling after analyzing the characteristics of cloud computing, for example, the communication of tasks and the choices of resources. A workflow task scheduling algorithm based on the resources' fuzzy clustering was proposed for shortening the completion time of task scheduling and improving task scheduling performance. This algorithm utilized the fuzzy clustering method to pretreat the resource and got the clustering results by depicting the resources comprehensive performance, so that narrowing the scale of resource choice and the processing unit time consumption. We used the classic HEFT, DLS algorithms as the benchmarks experiments. Based on our performance evaluation study and comparisons of these algorithms, the results indicated that FCBWTS algorithm had better performance and operability.

As the experiment environment was a simple heterogeneous environment without considering the dynamic changes of communication ability, FCBWTS algorithm applied the resource network structure in relatively stable state. The future research focuses on the improvement and expansion of the security on dynamic scheduling. At the same time, adding an effective security mechanism to ensure the safety of workflow task scheduling is very important for further research.

ACKNOWLEDGEMENTS

This research was supported by the Natural Science Foundation of Xinjiang Uygur Autonomous Region (No. 2013211A010) and the National Natural Science Foundation of China (No. 61063042, 61262088).

REFERENCES

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4):50–58.
2. Chen K, Zheng WM. Cloud computing: system instances and current research. *Journal of Software* 2009; **20**(5):1337–1348.
3. Zhang JX, Gu XM, Zheng C. Survey of research progress on cloud computing. *Application Research of Computers* 2010; **27**(2):429–433.
4. Rimal BP, Choi E. A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing. *International Journal of Communication Systems* 2012; **25**:796–819.
5. Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop* (*GCE'08*), 2008; 1–10. DOI: 10.1109/GCE.2008.4738445.
6. Papazchos ZC, Karatza HD. Scheduling of frequently communicating tasks. *International Journal of Communication Systems* 2012; **25**:146–157.
7. Luo H, Fan Y, Wu C. Over view of workflow technology. *Journal of Software* 2000; **11**(7):899–907.
8. Kaur N, Aulakh TS, Cheema RS. Comparison of workflow scheduling algorithms in cloud computing. *International Journal of Advanced Computer Science and Applications* 2011; **2**(10):81–86.
9. Wu M, Gajski D. Hypertool: a programming aid for message passing systems. *IEEE Transaction on Parallel and Distributed Systems* 1990; **1**(3):330−343.
10. Kwok YK, Ahmad I. Dynamic critical-path scheduling: an effective technique for allocating task graphs onto multiprocessors. *IEEE Transaction on Parallel and Distributed Systems* 1996; **7**(5):506−521.
11. Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transaction on Parallel and Distributed Systems* 2002; **13**(3):260–274.
12. Sih GC, Lee EA. A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures. *IEEE Transaction on Parallel and Distributed Systems* 1993; **4**(2):75–87.

DOI: 10.1002/dac

13. Yu J, Buyya R. A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. Workshop on workflows in support of large-scale Science, *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006),* Paris, France, June 19-23, 2006.
14. Braun TD, Siegel HJ, Beck N. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing system. *Journal of Parallel and Distributed Computing* 2001; **61**:810–837.
15. Zhi Q, Jiang C. A scheduling algorithm suitable for heterogeneous computing environment. *Acta Automatica Sinica* 2005; **31**(6):865–872.
16. Daoud MI, Kharma N. A high performance algorithm for static task scheduling in heterogeneous distributed computing system. *Journal of Parallel and Distributed Computing* 2008; **68**(4):399–409.
17. Lan Z, Sun SX. Scheduling algorithm based on critical tasks in heterogeneous environments. *Journal of Systems Engineering and Electronics* 2008; **19**(2):398–404.
18. Zhang M, Yu J. Fuzzy partitional clustering algorithms. *Journal of Software* 2004; **15**(6):858–868.
19. Zadeh LA. Is there a need for fuzzy logic. *Information Sciences* 2008; **178**(13):2751–2779.
20. Chen S, Li J, Wang X. *Fuzzy Set Theory and its Application*. Science Press: Beijing, 2005.
21. Zhang Y, Ma P, Su X, Zhang C. Multi-attribute decision making with uncertain attribute weight information in the framework of interval-valued intuitionistic fuzzy set. *Acta Automatica Sinica* 2012; **38**(2):220–228.
22. Buyya R, Yeo CS, Venugopal S. Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications* 2008; 5–13.
23. Wang LZ, Tao J, Kunze M. Scientific cloud computing: early definition and experience. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications* 2008; 825–830.
24. Cheng B. Hierarchical cloud service workflow scheduling optimization schema using heuristic generic algorithm. *PRZEGLAD ELEKTROTECHNICZNY* (*Electrical Review*) 2012; 92–95.

DOI: 10.1002/dac