INFORMATICA, 2015, Vol. 26, No. 1, 159–180

DOI: http://dx.doi.org/10.15388/Informatica.2015.43

Trust-Based Scheduling Strategy for Cloud Workflow Applications

Yuli YANG[1,2], Xinguang PENG[1] *, Jianfang CAO[1]

[1]*College of Computer Science and Technology, Taiyuan University of Technology*
No. 79, Yinze West Street, Taiyuan 030024, China
[2]*Department of Public Computer Teaching, Yuncheng University*
No. 1155, Fudan West Street, Yuncheng 044000, China
e-mail: yangyuliyyl@126.com, sxgrant@126.com, kcxdj122@126.com

Received: January 2014; accepted: October 2014

Abstract. Traditional researches on scheduling of cloud workflow applications were mainly focused on time and cost. However, security and reliability have become the key factors of cloud workflow scheduling. Taking time, cost, security and reliability into account, we present a trust-based scheduling strategy. We firstly formulate the cloud workflow scheduling and then propose the corresponding algorithm, in which the trustful computation service and storage service are selected according to the set-based particle swarm optimization (S-PSO) method and set covering problem (SCP) tree search heuristic method, respectively. Finally, experimental results show that, compared with traditional methods, the proposed algorithm has better performance.

Key words: cloud computing, cloud workflow system, cloud workflow scheduling, data retrieval, trust utility value.

1. Introduction

A cloud workflow refers to the implementation of a workflow application in a cloud computing environment. As a newly emerging paradigm in distributed computing, cloud computing can offer three different types of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). A cloud workflow management system can be deemed as PaaS (Foster *et al*., 2008), which is conducive to the automation of distributed applications, such as large-scale e-commerce and e-science (Bessis *et al*., 2013; Deelman *et al*., 2009). Executing the cloud workflow application has some potential benefits (Abrishami *et al*., 2013; Deelman, 2010). Firstly, the business model of pay-as-you-go makes users purchase computational and storage capacity according to their needs, thus reducing the cost of infrastructure (Choi *et al*., 2013). Secondly, the technology of virtualization adopted in a cloud computing environment allows the cloud workflow management system to create a fully customized execution environment for users. Meanwhile, the implementation of the cloud workflow application also gives rise to new

* Corresponding author.

challenges. For instance, because of the features of heterogeneity, openness and uncertainty in a cloud computing environment, how to select trustful computation services and storage services for workflow applications becomes an urgent problem (Wang *et al*., 2012).

The trust mechanism should be introduced in a cloud computing environment (Abbadi and Ruan, 2013). The concept of trust mentioned in the paper involves security and reliability. According to the users Quality of Service (QoS) requirements in time, cost, security and reliability, trustful scheduling of a cloud workflow application can choose appropriate computation services and storage services to fulfill tasks of the workflow application, even in an unstable network environment.

This paper is focused on the scheduling problem of cloud workflow applications composed of interdependent tasks: to retrieve of data sets required by each task from storage resources and to schedule the task on a computation service for implementation. In our previous work (Yang and Peng, 2013), we proposed a trust-based model of cloud workflow scheduling, which was focused on a trade-off between time and trust using a local search algorithm. Considering that the proposed model is not suitable for the pay-on-demand model in cloud computing and that the local search algorithm has weak search ability, we propose the improved model in this paper, in which the cost has been added and a global search algorithm has been used to find the approximate scheduling solution. Our contribution includes two aspects. Firstly, we formulate the problem of cloud workflow applications based on the comprehensive consideration of time, cost and trust constraints. Secondly, we propose a novel trust-based scheduling algorithm, in which the trustful computation services and storage services are selected according to the S-PSO and the SCP tree search heuristic algorithm, respectively.

This paper is organized as follows. Related research on workflow scheduling algorithms is reviewed in the next section. The scheduling model based on trust is formulated in Section 3. The trustful scheduling algorithm is described in Section 4. Simulation experiment and discussion are presented in Section 5 and conclusions are given in Section 6.

2. Related Work

The scheduling problem of a workflow application in a distributed computing environment has attracted a lot of attention from researchers. Venugopal and Buyya (2008) established a scheduling model for data-intensive applications and proposed the SCP tree search strategy to solve the model. Al-Mistarihi and Yong (2009) studied the way to ensure fairness when users selected a resource replica under the conditions of limited resources.

With the development of distributed data-intensive computing, in order to meet the QoS requirements of different users, the integration of reliability and security of scheduling algorithms had been studied. Wang *et al*. (2009) proposed a scheduling scheme, in which the trust mechanism was integrated into the life cycle of a scientific workflow to improve the predictability and robustness of the whole scheduling process. Kołodziej and Xhafa (2011) presented a scheduling scheme based on game theory and genetic algorithms for the scheduling problem with the security requirements. They also considered

the multi-objective scheduling problems including the reliability and security requirements (Kołodziej and Xhafa, 2012). Wang *et al*. (2012) proposed a scientific workflow scheduling model based on the trust mechanism described with Bayesian Theory. Yang and Peng (2013) built a cloud workflow scheduling model based on the comprehensive consideration of time and trust, and presented a local search algorithm to solve the model.

Moreover, since the workflow scheduling is a NP-Complete problem (Garey and Johnson, 1979), many meta-heuristic algorithms have been designed to solve cloud workflow scheduling problems: evolutionary algorithm (EA) (Zhu and Wang, 2008), ant colony optimization (ACO) (Chen and Zhang, 2009), and particle swarm optimization (PSO) (Pandey *et al*., 2010), in which rules are integrated with randomness to imitate natural phenomena and find the approximate solution by iteration.

Traditional researches on cloud workflow application scheduling were mainly focused on the optimization constrained by time and cost, but the trustful scheduling including security and reliability was seldom considered. In this paper, based on our previous work, in which a local search algorithm was used to make a trade-off between time and trust, we extend the approach and try to make it more suitable for cloud workflow scheduling problems. We formulate the cloud workflow scheduling problem by adding the cost factor, which is important in the cloud computing environment. Moreover, we propose a combined global search algorithm of S-PSO and SCP to obtain the approximate solution.

3. Statement of the Problem

Before formulating the scheduling problem for cloud workflow applications, we first describe the environment components of implementing a cloud workflow, and then introduce the requirements of security and reliability. Finally, the trust constraint model is also given for the workflow scheduling problem.

3.1. *Environment Descriptions*

The IaaS service provider offers customers the virtualized services including computation services and storage services. During the implementation of the cloud workflow application, it is usually required to store or transfer large amounts of data. For the cloud workflow application, we assume that the IaaS service provider offers computation services to run tasks and data storage services attached to the computation services. Amazon Elastic Compute Cloud (EC2)[2] and Amazon Elastic Block Store (EBS)[3] are respectively the representatives of the above two services. They are deployed in different data centers $DC = \{dc_1, dc_2, \ldots, dc_Q\}$ connected by different bandwidths. Figure 1 shows a simplified cloud computing environment consisting of five data centers. The numbers alongside the connecting links represent the bandwidths between different data centers.

[2]http://aws.amazon.com/ec2/.
[3]http://aws.amazon.com/ebs/.

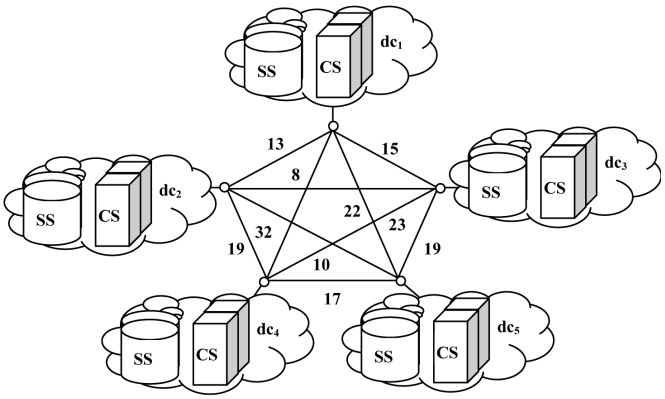

Fig. 1. A simplified cloud computing environment.

- **Computation service:** Suppose that there are M computation services denoted as $CS = \{cs_1, cs_2, \ldots, cs_M\}$ with corresponding computing capacity vector $C = \{c_1, c_2, \ldots, c_M\}$, where cs_i $(1 \leqslant i \leqslant M)$ can provide its computation service with security level S_{cs_i} and reliability level R_{cs_i}. $M_L = \{1, 2, \ldots, M\}$ is used to denote the set of computation services' labels.
- **Storage service:** Suppose that there are P available storage services denoted as $SS = \{ss_1, ss_2, \ldots, ss_P\}$, where ss_i $(1 \leqslant i \leqslant P)$ can offer storage service with security level S_{ss_i} and reliability level R_{ss_i}.
- **Workflow application**: The workflow application can be modeled as a directed acyclic graph DAG $G = \langle T, V \rangle$, where T is the set of N tasks $\{t_1, t_2, \ldots, t_N\}$; an edge $(t_k, t_i) \in V$ indicates the constraint relationship between task t_k and t_i: t_i could not be executed unless t_k is performed and the result is returned to t_i, the immediate predecessors t_k of task t_i is denoted as $t_k \in pred(t_i)$. Each task $t_i \in T$ has input data, communication data, and output data, which are usually stored as data files. t_i's processing length is denoted as l_i. Assume that the ith task requires a set of K_i input data sets denoted by F^{t_i}, which are distributed on a subset of storage services. Especially, for a data set $f \in F^{t_i}$, the storage service which stores a copy of f is marked as ss_f. The requirements of security and reliability of task t_i are respectively denoted as SD_{t_i} and RD_{t_i}. The communication data are the amount of data transmitted from $t_k \in pred(t_i)$ to task t_i. An entry task is the task without any parent and an exit task is the task without any child. Assume that the DAG in the paper has a single entry and exit task. As shown in Fig. 2 (Yang and Peng, 2013), this scenario depicts a workflow application including ten interdependent tasks. Taking the ninth task as an example, communication data are stored as data files, and transferred from computation services used for executing immediate predecessors task t_4 and t_5, to the computation service used for executing t_9. Besides, t_9 requires 3 input data sets f_1, f_2 and f_3, which are replicated on different storage services. The adjacency matrix $A_9 = [a_{jk}]$ $(1 \leqslant j \leqslant P,\ 1 \leqslant k \leqslant K_9)$ is used to represent the storages of data sets, where $a_{jk} = 1$ if the kth data set f (denoted as f_k) is replicated on storage service

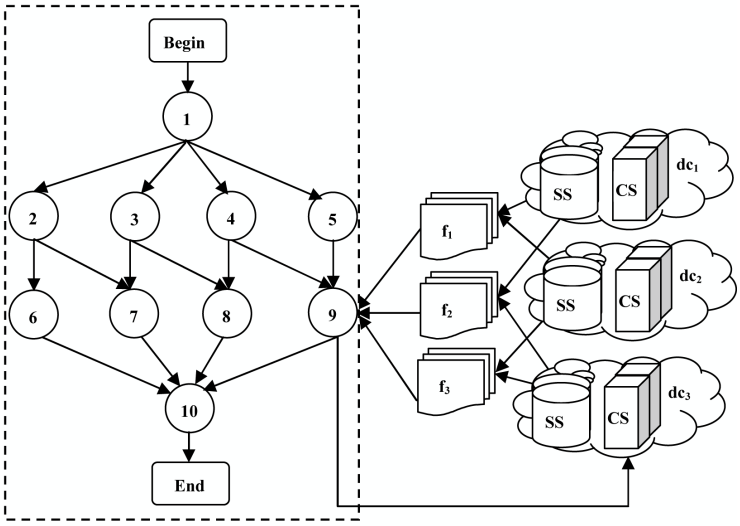

Fig. 2. A DAG of workflow application.

$$A_9 = \begin{array}{c} \begin{matrix} \mathrm{f}_1 & \mathrm{f}_2 & \mathrm{f}_3 \end{matrix} \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ \hdashline 1 & 0 & 1 \\ \hdashline 0 & 1 & 1 \end{array}\right] \end{array} \begin{array}{l} \mathrm{dc}_1 \\ \mathrm{dc}_2 \\ \mathrm{dc}_3 \end{array}$$

$$Tab_9 = \begin{array}{c} \begin{matrix} \mathrm{f}_1 & \mathrm{f}_2 & \mathrm{f}_3 \end{matrix} \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ \hdashline 1 & 1 & 0 \\ 0 & 1 & 1 \\ \hdashline 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right] \end{array} \begin{array}{l} \mathrm{dc}_1 \\ \mathrm{dc}_2 \\ \mathrm{dc}_1 \\ \mathrm{dc}_3 \\ \mathrm{dc}_2 \\ \mathrm{dc}_3 \end{array}$$

Fig. 3. The adjacency matrix A_9 and tableau Tab_9 of task t_9 in Fig. 2.

ss_f ($f \in F^{t_9}$); P is the number of storage services; K_9 is the number of data sets F^{t_9} required by t_9; Tableau Tab_9 consisting of K_9 blocks is created for obtaining all storage service solution sets according to the adjacency matrix A_9. The row number contained in the kth ($1 \leqslant k \leqslant K_9$) block is the number of storage services that contain the kth data set f, and the contained rows have the same sorting order with the rows in A_9. The adjacency matrix A_9 and tableau Tab_9 of task t_9 in Fig. 2 are shown in Fig. 3 (Yang and Peng, 2013).

- **Resource set:** The selected computation service and storage services are collectively referred as the resource set regarding the ith task, and expressed as $S^i = \{cs^i, \{ss^i\}\}$, where $cs^i = cs_j$ ($cs_j \in CS$) represents the computation service selected for executing the ith task and $ss^i \subseteq \cup_{f \in F^{T_i}} ss_f$ is the set of storage services chosen for obtaining the data sets required by the ith task.

Table 1
Security level, concept and range.

Security level	Security concept	Security range
1	Insecurity	[0.0, 0.2)
2	Low security	[0.2, 0.4)
3	Medium security	[0.4, 0.6)
4	Very security	[0.6, 0.8)
5	High security	[0.8, 1.0]

3.2. *Security and Reliability Requirements*

In order to quantify the trust relationship between task t_i and resource set $S^i = \{cs^i, \{ss^i\}\}$, the matrix $P_{ss}[ss_f][t_i]$ is used to denote the failure probability of storage service ss_f chosen for transferring the data set f required by task t_i ($f \in F^{t_i}$). The matrix elements are explained as the failure probabilities of ss_f during transferring data set f because of the high security restrictions (Kołodziej and Xhafa, 2011). The probability matrix $P_{ss}[ss_f][t_i]$ is calculated as Eq. (1):

$$P_{ss}[ss_f][t_i] = \begin{cases} 0, & SD_{t_i} \leqslant S_{ss_f} \\ 1 - e^{-\lambda(k_1 - k_2)}, & SD_{t_i} > S_{ss_f} \end{cases} \tag{1}$$

where the security demand vector of task t_i is denoted as SD_{t_i} ($i = 1, \ldots, N$), and S_{ss_f} is the security level vector of storage service ss_f. The values of different security levels, corresponding security concepts, and security ranges are shown in Table 1 (Yang and Peng, 2013). According to the results by Song *et al.* (2005), a real fraction in the range [0, 1] is used to quantified customers security demand and the security level of resources. The higher the value is, the higher the users security demand or the security level of resources is. Based on the results, in order to quantify them more finely, we divide [0, 1] into five subintervals with the same length according to five security levels. The values of SD_{t_i} and S_{ss_f} are integers from 1 to 5. The parameter λ is interpreted as a failure coefficient and is set to be 3 according to Song *et al.* (2006), k_1 and k_2 are random values in the security range of the corresponding security level.

Under some circumstances, the special strategies of storage services supplier or the system's dynamic character may make the storage service unavailable (Kołodziej and Xhafa, 2012). Therefore, the storage service ss_f for transferring data set f required by task t_i may be no longer available with a certain probability $P_{sr}[ss_f][t_i]$ calculated by Eq. (2).

$$P_{sr}[ss_f][t_i] = 1 - P_r[ss_f] \tag{2}$$

where the reliability probability (Rood and Lewis, 2008) $P_r[ss_f]$ is a random value in the range [0, 1] and determined by the reliability level of the storage service ss_f. The higher the value $P_r[ss_f]$ is, the higher the reliability of resources is. Like the method of quantifying security, we divide [0, 1] into five subintervals with the same length to

Table 2
Reliability level, concept and probability.

Reliability level	Reliability concept	Reliability probability
1	Unreliability	[0.0, 0.2)
2	Low reliability	[0.2, 0.4)
3	Medium reliability	[0.4, 0.6)
4	Very reliability	[0.6, 0.8)
5	High reliability	[0.8, 1.0]

quantify the reliability probability more finely according to five reliability levels. The values of different reliability levels are shown in Table 2 (Yang and Peng, 2013).

Similar to storage services, the failure probabilities owing to the trust relationships between computation service cs_j and task t_i are denoted as $P_{cs}[cs_j][t_i]$ and $P_{cr}[cs_j][t_i]$, respectively. The cloud workflow scheduling model extended using $P_{ss}[ss_f][t_i]$, $P_{sr}[ss_f][t_i]$, $P_{cs}[cs_j][t_i]$ and $P_{cr}[cs_j][t_i]$ is presented in the following subsection.

3.3. *Trustful Cloud Workflow Scheduling Model*

Traditional cloud workflow scheduling model contains the following two objectives:

- **Time minimization**: Time indicates the response speed of cloud service request. It should spend the least time to complete all tasks of the cloud workflow application.
- **Cost minimization**: Cost is the total expenses of completing all tasks and should be minimized.

Supposing that task $t_i \in T$ is assigned to computation service $cs_t \in CS$ for execution with non-preemptive approach. $Time(t_i)$ is defined as the completion time of task t_i and is composed of two parts: one is the time required for obtaining communication data and input data sets before executing task t_i; the other is the time required for executing t_i on cs_t and denoted by $T_e(t_i, cs_t)$. The former is computed by $\mathrm{Max}[\mathrm{Max}_{t_k \in pred(t_i)}(Time(t_k) + T_d(t_k, t_i)), \mathrm{Max}_{f \in F^{t_i}}(T_n(f, ss_f, cs_t), avail(cs_t)]$, in which $Time(t_k)$ is the time of implementing the immediate predecessor $t_k \in pred(t_i)$; $T_d(t_k, t_i)$ is the time required for transferring communication data from task t_k to task t_i; the time required for obtaining communication data is the maximum time of task $t_k \in pred(t_i)$ completing the data transmission process and denoted as $\mathrm{Max}_{t_k \in pred(t_i)}(Time(t_k) + T_d(t_k, t_i))$. $T_n(f, ss_f, cs_t)$ is the time required for transferring the required data sets f retrieved from selected storage service ss_f to assigned computation service cs_t. The time required for obtaining data sets $f \in F^{t_i}$ in parallel can be computed by $\mathrm{Max}_{f \in F^{t_i}}(T_n(f, ss_f, cs_t)$. Besides, $avail(cs_t)$ is the earliest time when computation service cs_t is ready for executing task t_i. Therefore, $Time(t_i)$ can be calculated by Eq. (3)

$$Time(t_i) = \mathrm{Max}\Big[\max_{t_k \in pred(t_i)}\big(Time(t_k) + T_d(t_k, t_i)\big), \max_{f \in F^{t_i}}\big(T_n(f, ss_f, cs_t)\big), avail(cs_t)\Big] + T_e(t_i, cs_t). \tag{3}$$

Supposing that task t_k is assigned to computation service cs_v for execution, then the time required for transferring communication data from task t_k to task t_i is defined as $T_d(t_k, t_i) = data_{t_k,t_i}/BW(dc_{cs_v}, dc_{cs_t})$. $data_{t_k,t_i}$ represents the amount of communication data transmitted from task t_k to task t_i. $BW(dc_{cs_v}, dc_{cs_t})$ is the bandwidth between the data centers respectively containing computation services cs_v and cs_t. $T_d(t_k, t_i)$ becomes zero when both task t_k and t_i are run on the same computation service. The time $T_n(f, ss_f, cs_t)$ is estimated using Eq. (4)

$$T_n(f, ss_f, cs_t) = T_w(ss_f) + Size(f)/BW(dc_{ss_f}, dc_{cs_t}), \tag{4}$$

where $T_w(ss_f)$ is the waiting time from making the request to receiving the first byte of the data set f; $Size(f)$ represents the amount of data set f; $BW(dc_{ss_f}, dc_{cs_t})$ represents the bandwidth between data centers containing storage service ss_f and computation service cs_t. The desired execution time $T_e(t_i, cs_t)$ of task t_i is given by Eq. (5)

$$T_e(t_i, cs_t) = l_i/c_t, \tag{5}$$

where l_i and c_t denote the processing length of task t_i and the computing capacity of computation service cs_t, respectively.

Let $Cost(t_i)$ be the economic cost of implementing task t_i, which includes the transfer costs of communication data from all immediate predecessors $t_k \in pred(t_i)$ of task t_i, the data retrieving cost of data sets from the selected storage service, and the execution cost of task t_i on assigned computation service. It is calculated by Eq. (6)

$$Cost(t_i) = \sum_{t_k \in pred(t_i)} C_c(data_{t_k,t_i}) + \sum_{f \in F^{t_i}} C_d(f) + C_e(t_i, cs_t), \tag{6}$$

where $C_c(data_{t_k,t_i})$ is used to compute the transfer cost of communication data transmitted from the immediate predecessor task t_k to task t_i, and it is determined by Eq. (7); $data_{t_k,t_i}$ denotes the amount of communication data transmitted from task t_k to task t_i; $Price(cs_k, cs_t)$ denotes the transfer cost of unit data from computation service cs_k to cs_t

$$C_c(data_{t_k,t_i}) = data_{t_k,t_i} * Price(cs_v, cs_t). \tag{7}$$

The transfer cost of data set f from the selected storage service ss_f to the allocated computation service cs_t is computed by Eq. (8)

$$C_d(f) = Size(f) * Price(ss_f, cs_t), \tag{8}$$

where $Size(f)$ is the amount of data set f, and $Price(ss_f, cs_t)$ denotes the transfer cost of unit data from storage service ss_f to computation service cs_t. The execution cost of task t_i on computation service cs_t is computed by Eq. (9)

$$C_e(t_i, cs_t) = T_e(t_i, cs_t) * Price(cs_t), \tag{9}$$

where $T_e(t_i, cs_t)$ is the execution time of task t_i, and $Price(cs_t)$ is the cost of computation service cs_t per unit time.

According to the completion time of task t_i, Eq. (10) is used to compute the makespan of the cloud workflow application modeled as DAG

$$Makespan(DAG) = \mathrm{Max}_{t_i \in T}\left(Time(t_i)\right). \tag{10}$$

Similarly, the total cost of the cloud workflow application is defined as Eq. (11)

$$Cost(DAG) = \sum_{t_i \in T} Cost(t_i). \tag{11}$$

The heterogeneous, dynamic and open characteristics of cloud computing environments bring a lot of uncertainty on the execution of the workflow application. How to obtain trustful service becomes a critical issue. For the purpose of integrating the trust relationship between resource sets and tasks into the calculation process of time and cost, the scheduling model is extended by considering the failure probabilities and the unavailable probabilities of selected resource sets. So, the completion time $Time(t_i)$ of task t_i can be calculated in three steps. Firstly, considering the failure probabilities and the unavailable probabilities of computation service cs_t and cs_v caused by transferring communication data between them, which are denoted as $P_{cs}[cs_t][t_i]$, $P_{cr}[cs_t][t_i]$, $P_{cs}[cs_v][t_i]$ and $P_{cr}[cs_v][t_i]$, respectively, the expected time of gaining communication data can be extended based on $T_d(t_k, t_i)$ and calculated using Eq. (12). Secondly, the failure probabilities and the unavailable probabilities of storage service ss_f and computation service cs_t, which are denoted as $P_{ss}[ss_f][t_i]$, $P_{sr}[ss_f][t_i]$, $P_{cs}[cs_t][t_i]$ and $P_{cr}[cs_t][t_i]$, are considered when estimating the expected time of retrieving data set $f \in F^{t_i}$. It can be extended based on $T_n(f, ss_f, cs_t)$ and computed by Eq. (13). Thirdly, the failure probability and the unavailable probability of computation service cs_t denoted as $P_{cs}[cs_t][t_i]$ and $P_{cr}[cs_t][t_i]$ are considered for estimating the expected execution time, which can be extended based on $T_e(t_i, cs_t)$ and estimated by Eq. (14). Based on Eqs. (12) to (14), the completion time $Time'(t_i)$ can be computed by Eq. (15):

$$T'_d(t_k, t_i) = T_d(t_k, t_i) * \left(1 + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i] + P_{cs}[cs_v][t_i] + P_{cr}[cs_v][t_i]\right), \tag{12}$$

$$T'_n(f, ss_f, cs_t) = T_n(f, ss_f, cs_t) * \left(1 + P_{ss}[ss_f][t_i] + P_{sr}[ss_f][t_i] + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i]\right), \tag{13}$$

$$T'_e(t_i, cs_t) = T_e(t_i, cs_t) * \left(1 + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i]\right), \tag{14}$$

$$Time'(t_i) = \mathrm{Max}[\mathrm{Max}_{t_k \in pred(t_i)}\left(Time'(t_k) + T'_d(t_k, t_i)\right), \mathrm{Max}_{f \in F^{t_i}}\left(T_n(f, ss_f, cs_t)\right), avail(cs_t)\big] + T'_e(t_i, cs_t). \tag{15}$$

So, the makespan of the workflow application is computed by Eq. (16)

$$Makespan'(DAG) = \mathrm{Max}_{t_i \in T}\left(Time'(t_i)\right). \tag{16}$$

Similar to the makespan of the workflow application, the cost of it can be extended by Eqs. (17), (18), (19), (20) and (21):

$$C'_c(data_{t_k,t_i}) = C_c(data_{t_k,t_i}) * \big(1 + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i] + P_{cs}[cs_v][t_i] + P_{cr}[cs_v][t_i]\big), \tag{17}$$

$$C'_d(f) = C_d(f) * \big(1 + P_{ss}[ss_f][t_i] + P_{sr}[ss_f][t_i] + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i]\big), \tag{18}$$

$$C'_e(t_i, cs_t) = C_e(t_i, cs_t) * \big(1 + P_{cs}[cs_t][t_i] + P_{cr}[cs_t][t_i]\big), \tag{19}$$

$$Cost'(t_i) = \sum_{t_k \in pred(t_i)} C'_c(data_{t_k,t_i}) + \sum_{f \in F^{t_i}} C'_d(f) + C'_e(t_i, cs_t), \tag{20}$$

$$Cost'(DAG) = \sum_{t_i \in T} Cost'(t_i). \tag{21}$$

According to the objectives of time and cost with different dimensions, the method of normalization described in Eq. (22) can be used to map each objective into a value between 0 and 1 and translate the multi-objective optimization into a weighted sum of objectives. Supposing that there are Q cloud workflow scheduling solutions, for a certain solution S_i $(1 \leqslant i \leqslant Q)$, the value of which can be computed according to Eq. (22), and be denoted as the comprehensive utility value. The scheduling solution that corresponds to the maximum comprehensive utility value is deemed as the final solution

$$ComU(S_i) = w_1 \frac{T_{\max} - T'(S_i)}{T_{\max} - T_{\min}} + w_2 \frac{C_{\max} - C'(S_i)}{C_{\max} - C_{\min}}. \tag{22}$$

In Eq. (22), $T'(S_i)$ and $C'(S_i)$ are denoted the makespan and the cost of a certain solution S_i $(1 \leqslant i \leqslant Q)$. $T_{\max}(T_{\min})$ is the maximal (minimal) makespan of all scheduling solutions; $C_{\max}(C_{\min})$ is the maximal (minimal) cost of all scheduling solutions. After calculating the makespan and cost of all scheduling solutions and selecting the maximal (minimal) makespan and maximal (minimal) cost, $T_{\max}(T_{\min})$ and $C_{\max}(C_{\min})$ can be obtained. $w_k \in R^+(\sum_{k=1}^{2} w_k = 1)$ is a non-negative weight of the kth objective function. Assuming that the security and reliability are equally important in this paper, $w_1 = 0.5$ and $w_2 = 0.5$ are adopted in this paper.

4. Trust-Based Heuristic Scheduling Algorithm

In this section, we present a trust-based heuristic scheduling algorithm (TBHSA) for the cloud workflow application. The optimization process of TBHSA consists of two parts. The steps of mapping the relationship between tasks and computation services with S-PSO are introduced in Algorithm 1. The steps of mapping the task-storage services relationship with SCP tree search heuristic method are listed in Algorithm 2.

4.1. *S-PSO for Task-Computation Service Mapping*

The trust-based scheduling of cloud workflow applications belongs to a typical discrete optimization problem (Fabio *et al*., 2009). Therefore, we attempt to establish the mapping relationship between tasks and computation services with a discrete PSO algorithm named S-PSO. To understand S-PSO well, the standard discrete particle swarm optimization is introduced firstly as follows.

4.1.1. *Canonical Model*

The canonical PSO is a self-adaptive stochastic intelligence algorithm developed by Kennedy and Eberhart (1995) under the inspiration of the group behaviors of many birds. The process when the birds fly to the habitat with a good food source can be interpreted as the process to find an approximate solution according to a set of rules. Each individual in PSO is considered as a particle without quality or volume, which can fly in the N-dimension space at a certain speed to search for better solutions. Each particle adjusts its speed and direction of the flight dynamically according to the flying experience of individuals and groups. The updating rules of the particle's velocity and position are shown in Eqs. (23) and (24):

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 \left(pbest_i - X_i^t\right) + c_2 r_2 (gbest - X_i^t), \tag{23}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, \tag{24}$$

where V_i^t and X_i^t are the velocity and position of particle i at the tth iteration, respectively; $pbest_i$ is the best position of particle i, which has the best fitness value computed according to the fitness function; $gbest$ is the position of the best particle in the whole swarm; ω is the inertia weight; c_1 and c_2 represent acceleration factors of individual and social perception; r_1 and r_2 are the random values between 0 and 1 to maintain the diversity of the population.

4.1.2. *The S-PSO*

Because the updating rules (23) and (24) are all defined in a continuous space, they cannot be used to solve optimization problems in the discrete space directly. We adopt the S-PSO (Chen *et al*., 2010) to find a suitable mapping between tasks and computation services. Lin and Kernighan (1973) believed that a combinatorial optimization problem (COP) usually could be formulated as such a model, in which a subset $X = X^1 \cup X^2 \cup \cdots \cup X^N$ was found from a universal set $E = E^1 \cup E^2 \cup \cdots \cup E^N$ divided into N dimensions to satisfy certain constraints Ω and optimize the objective function f simultaneously. Based on the point above, we define the search space of the cloud workflow with N tasks as a universal set $E = E^1 \cup E^2 \cup \cdots \cup E^N$, where $E^i = \{cs_i^1, cs_i^2, \ldots, cs_i^r\}$ represents r available computation services for t_i. Meanwhile, a scheduling solution of the cloud workflow is given as $X_i = (X_i^1, X_i^2, \ldots, X_i^N)$, in which $X_i^j \in E^j$ $(j = 1, 2, \ldots, N)$ is the number of computation service used to execute task t_j. X_i is a feasible solution only if it satisfies cloud customers' QoS requirements. The goal of the cloud workflow scheduling problem is to find an approximate solution $X_i^* \subseteq E$, which owns the maximum comprehensive

utility value according to Eq. (22). The updating rule (23) is used by the S-PSO under the premise of operations in the set space. The redefined operators are given as follows:

- **Position:** A position in S-PSO denotes a feasible solution of the cloud workflow scheduling problem. Namely, the position $X_i = (X_i^1, X_i^2, \ldots, X_i^N)$ $(X_i \subseteq E)$ of particle i is composed of N dimensions. For each dimension, $X_i^j \in E^j$ $(j = 1, 2, \ldots, N)$. $pbest_i \subseteq E$ and $gbest \subseteq E$ represent the best position of particle i and the population.
- **Velocity:** The velocity $V_i = \{e/p(e)|e \in E\}$ of particle i is a set defined on E with the probability $p(e)$ and each dimension $V_i^j = \{e/p(e)|e \in E^j\}$ in velocity V_i is a set defined on E^j with the probability $p(e)$.
- **Coefficient × velocity:** Supposing that coefficient c is a non-negative real number, the result of c multiplied by velocity V is defined as Eq. (25)

$$cV = \left\{e/p'(e)|e \in E\right\}, \quad p'(e) = \begin{cases} 1, & \text{if } c \times p(e) > 1, \\ c \times p(e), & \text{otherwise.} \end{cases} \tag{25}$$

- **Position − position:** The subtraction operation between two positions is defined as Eq. (26).

$$A - B = \{e|e \in A \text{ and } e \notin B\}. \tag{26}$$

- **Coefficient × (position – position):** The operator about "Coefficient × (position – position)" is defined as Eq. (27)

$$cE' = \left\{e/p'(e)|e \in E\right\}, \quad p'(e) = \begin{cases} 1, & \text{if } e \in E' \text{ and } c > 1, \\ c, & \text{if } e \in E' \text{ and } 0 \leqslant c \leqslant 1, \\ 0, & \text{if } e \notin E'. \end{cases} \tag{27}$$

- **Velocity + velocity:** Assuming that velocity V_1 and V_2 are the sets defined on E with the probability $p_1(e)$ and $p_2(e)$, respectively. The addition operation with V_1 and V_2 is defined as Eq. (28)

$$V_1 + V_2 = \left\{e/\max(p_1(e), p_2(e))|e \in E\right\}. \tag{28}$$

The updating rules of the particle's position in the discrete space include the following steps:

Step 1: The jth dimension of the velocity V_i is converted into a crisp set $cut_a(V_i^j) = \{e|e/p(e) \in V_i^j,\ p(e) \geqslant a\}$, where a is a random value between 0 and 1.

Step 2: Starting from an empty set, the position value of each dimension about the ith particle is established from the crisp set $cut_a(V_i^j)$, the previous position X_i^j, and other feasible elements successively.

Supposing that *PNum* denotes the size of particle swarm, then the number of particle's dimensions is equal to the number of tasks included in the cloud workflow application.

Algorithm 1 The S-PSO

Input:

$G = \langle T, E \rangle$: The DAG of the cloud workflow application
l_j $(1 \leqslant j \leqslant N)$: Task t_j's length
SD_{t_j} and RD_{t_j}: Task t_j's security and reliability demand
f $(f \in F^{t_j})$: A set of data sets required by task t_j
$Size(f)$: The size of data set f
ss_f: The storage service storied data set f
S_{ss_f} and R_{ss_f}: Storage service ss_f's security and reliability level
$_{sr}Price(cs_t)$ $(1 \leqslant t \leqslant M)$: The cost of computation service cs_t
S_{cs_t} and R_{cs_t}: Computation service cs_t's security and reliability level
$Price(UniDat)$: The cost of transferring unit data between resource sets
$PNum$: The size of particle swarm
g and G: The current iteration and the number of maximum iteration
$BW[dc_x][dc_y]$ $(1 \leqslant x \leqslant Q, 1 \leqslant y \leqslant Q)$: The bandwidth between data centers

Output:

Schedule

1: **for** $t_j \in T$ **do**
2: $Create(A_j)$ according to Section 2.1
3: $Compute(P_{ss}[ss_f][t_j], P_{sr}[ss_f][t_j], P_{cs}[cs_t][t_j], P_{cr}[cs_t][t_j])$ according to Eqs. (1), (2) et al.
4: **end for**
5: $Initialize(v_{ij}^1, x_{ij}^1)$ $(1 \leqslant i \leqslant PNum, 1 \leqslant j \leqslant N)$
6: $StorageServiceSelect(x_{ij}^1, A_i)$ according to Section 3.2
7: $Calculate(fitness(X_i))$ according to Eq. (22)
8: $Initialize(pbest_i^1, gbest^1)$
9: **while** $g < G$ **do**
10: $Update(v_{ij}^g)$ according to Eq. (23)
11: $Update(x_{ij}^g)$ according to Section 3.1.2
12: $StorageServiceSelect(x_{ij}^g, A_j)$ according to Section 3.2
13: $Update(fitness(X_i))$ according to Eq. (22)
14: $Update(pbest_i^g, gbest^g)$
15: $g{+}{+}$
16: **end while**
17: Output $gbest_j^g$ and corresponding storage services

The position value of each dimension is denoted the number of computation service used for executing the corresponding task. So each dimension's value of a particle is limited to computation services' labels $M_L = \{1, 2, \ldots, M\}$, namely, $X_i = \{x_{ij} | x_{ij} \in M_L,\ 1 \leqslant i \leqslant PNum,\ 1 \leqslant j \leqslant N\}$. Since a particle indicates a scheduling solution of the cloud workflow application, the terms "particle" and "solution" can often be used interchangeably. The

fitness function used to evaluate the performance of the particle is designed as Eq. (22). The S-PSO algorithm for mapping all tasks to a set of given computation services includes three different stages as follows:

Initialization stage (lines 1–8): Firstly, adjacency matrix A_j of task t_j is generated according to Section 2.1. Then, the probabilities of the failures caused by the security and reliability of resource sets are computed. In Line 5, the position of the particle x_{ij}^1 is initialized randomly with computation services' labels $M_L = \{1, 2, \ldots, M\}$, and the velocity of the particle v_{ij}^1 is initialized by selecting a positive integer from 1 to M and a random value in [0,1]. The mapping relationship between task t_j and storage services is described in Section 3.2. According to the resource sets $S^j = \{cs^j, \{ss^j\}\}$ of task t_j, each particle is evaluated by Eq. (22) in line 7. In the first iteration, the $pbest_i^1$ of particle i and the $gbest^1$ of the population are initialized by a copy of x_{ij}^1 and the best $pbest_s^1 (1 \leqslant s \leqslant PNum)$ among all the particles.

Execution stage (lines 9–16): In each iteration, the updating velocity of particle i is determined by Eq. (23) and all operators in Eq. (23) follow the redefinitions in Section 3.1.2. After updating the velocity of the particle, the particle's position is updated according to the updating steps of the position given in Section 3.1.2. The data sets required by the jth task are obtained through the storage service selection tactics introduced in Section 3.2. Then each particle's fitness value will be updated by Eq. (22), and $pbest_i^g$ and $gbest^g$ in the gth iteration will be updated according to the results. At last, g will be increased by 1 and the next iteration will be started.

Termination stage (line 17): Through the procedure outlined in Algorithm 1, the best solution chosen from all feasible solutions, which has the maximum comprehensive utility value according to Eq. (22), is deemed as the final solution.

4.2. *The SCP Tree Search Algorithm for Task-Storage Service Mapping*

The SCP tree search strategy was proposed by Venugopal and Buyya (2008) to map tasks to resources. In this paper, this strategy is improved through considering the trust relationship between resource sets and tasks. The steps in the improved algorithms for choosing storage services are listed in Algorithm 2.

It starts with initializing the following parameters (*line* 1), where B is used to keep the selected storage services. The set U contains the data sets already covered by the storage service solution set. All schemes of storage service solution sets are stored in B_{store}^{Num}, and Num is used to record the number of schemes. B_{Best}^j is used to store the best storage service set solution for task t_j and the variable z is the value calculated by the current solution set. Tableau Tab_j is generated according to Section 2.1 (*line* 2). Then, the blocks in Tab_j are searched sequentially to obtain all storage service solution sets (*line* 3 and *lines* 9–18), which starts from the smallest index k ($f_k \notin U$). In the kth block, d_q^k indicates the storage service ss_k, where q is a row pointer in block k. If the row contains 1, corresponding data set f_i should be added to U and corresponding storage service d_q^k should be added to B. If all the data sets have been covered ($U = F^{t_j}$), B is stored in B_{store}^{Num} and Num is updated; otherwise, a recursive method is adopted to obtain all the

Algorithm 2 *StorageServiceSelect*(cs^j, A_j)

Input:

The adjacency matrix A_j of task t_j

Output:

The storage services B^j_{Best} of task t_j

1: *Initialize*($B = \emptyset,\ U = \emptyset,\ B^{Num}_{store} = \emptyset,\ Num = 0, B^j_{Best} = \emptyset,\ z = 0$)
2: *Create*(Tab_j)
3: *Search*(t_j, B, Tab_j, U)
4: **for** $i \in [1, Num]$ **do**
5: if $z <$ *fitness*(CS^j, B^i_{store}) then
6: $B^j_{Best} \leftarrow B^i_{store}, z \leftarrow$ *fitness*(CS^j, B^i_{store})
7: **end for**
8: *Output*(B^j_{Best})
9: *Search*(t_j, B, Tab_j, U)
10: *Mark*(Tab^k_j)($f_k \notin U$)
11: **while** $q \leqslant Tab^k_j$ **do**
12: $F_T \leftarrow \{f_i | t_{qi} = 1,\ 1 \leqslant i \leqslant K\}$
13: $U \leftarrow U \cup F_T,\ B \leftarrow B \cup \{d^k_q\}$
14: if $U = F^{t_j}$ then
15: $B^{Num}_{store} \leftarrow B, Num++$
16: else *Search*(t_j, B, Tab_j, U)
17: $B \leftarrow B - \{d^k_q\}, U \leftarrow U - F_T, q++$
18: **end while**

solution sets in the branches of different blocks. At last, the best storage service solution set is selected from B^{Num}_{store} for task t_j (*lines* 4–8). For each storage service solution set B^i_{store} ($i \in [1, Num]$), if the fitness value calculated according to Eq. (22) is higher than the existing value of z, then B^j_{Best} is replaced by B^i_{store} and z is updated according to the fitness value.

4.3. *Experimental Settings*

To evaluate the proposed algorithm, we developed a simulation platform based on Cloudsim (2012). Due to the limited availability of data about cloud workflow scheduling, we have to use the data generated randomly to simulate the process. To understand the simulation process more clearly and orderly, we have divided parameters into three categories. The parameters and their value ranges contained in each category are given in Table 3.

We adopted makespan, cost, and trust utility metrics for performance evaluation. Makespan is the time of completing all the tasks of the cloud workflow application and can be computed by Eq. (16). The cost is the sum of the costs of completing all tasks and can be computed by Eq. (21). Suppose the resource sets selected for task t_i is denoted as

Table 3
Simulation parameter setting.

Category	Parameter	Range of vale
DAG	The number of tasks	[10, 100]
	The processing length of each task	[10, 100]
	The size of data files	[100, 1000]
Cloud platform	The bandwidth between data centers	[20, 50]
	Computation services' computing capacity	[2, 10]
	Resource sets' security and reliability	[1, 5]
	The cost of resource sets	[1, 3]
QoS requirement	Requirement of security and reliability	[1, 5]

$S^i = \{cs^i, \{ss^i\}\}$, where $cs^i = cs_t$ represents the computation service for executing task t_i and $ss^i \subseteq \cup_{f \in F^{T_i}} ss$ is the set of storage services for obtaining the required data sets. Security utility value $CSeU(cs_t, t_i)$ and reliability utility value $CReU(cs_t, t_i)$ of executing task t_i on computation service cs_t are respectively defined as Eq. (29) and Eq. (30), in which S_{cs_t} and R_{cs_t} represent the security and reliability level of computation service cs_t; SD_{t_i} and RD_{t_i} denote security and reliability demand of task t_i; the values of S_Max and R_Max denoted the maximal level of security and reliability demand are all equal to 5 according to Tables 1 and 2.

$$CSeU(cs_t, t_i) = \begin{cases} 1, & S_{cs_t} \geqslant SD_{t_i}, \\ 1 - \frac{SD_{t_i} - S_{cs_t}}{S\_\text{Max} - S_{cs_t}}, & S_{cs_t} < SD_{t_i}, \end{cases} \tag{29}$$

$$CReU(cs_t, t_i) = \begin{cases} 1, & R_{cs_t} \geqslant RD_{t_i}, \\ 1 - \frac{RD_{t_i} - R_{cs_t}}{R\_\text{Max} - R_{cs_t}}, & R_{cs_t} < RD_{t_i}. \end{cases} \tag{30}$$

The definitions of security utility value $SSeU(ss_f, t_i)$ and reliability utility value $SReU(ss_f, t_i)$ of obtaining data set $f \in F^{t_i}$ required by task t_i from storage service ss_f are similar to those of computation service cs_t. The trust utility value of task t_i can be defined as Eq. (31)

$$\begin{aligned} TUtil\left(t_i, cs_t, \left\{ss^i\right\}\right) = {} & CSeU(cs_t, t_i) + CReU(cs_t, t_i) \\ & + \sum_{f \in F^{t_i}} \left(SSeU(ss_f, t_i) + SReU(ss_f, t_i)\right). \end{aligned} \tag{31}$$

The total trust utility value of the cloud workflow is calculated by Eq. (32)

$$TotTUtil(DAG) = \sum_{t_i \in T} TUtil\left(t_i, cs_t, \left\{ss^i\right\}\right). \tag{32}$$

It should be pointed out that the trust of resource sets including security and reliability is integrated into the fitness function expressed as Eq. (22). In order to obtain maximum

comprehensive utility value, the scheduling scheme should allocate tasks to trustful resource sets, which can decrease makespan and cost to some extent. Therefore, the approximate solution should make a trade-off among makespan, cost and the total trust utility value.

4.4. *Simulation Experiments and Evaluations*

4.4.1. *Experimental Results*

First of all, a small-scale scheduling example is used to test the performance of TBHSA. It includes a workflow application with ten tasks, six computation services and six storage services, denoted as (T10, CS6, SS6). The DAG model of the workflow application is shown in Fig. 2. The performances of TBHSA, SCP-based scheduling algorithm (SCPSA) (Venugopal and Buyya, 2008), and trust-based scheduling algorithm (TSA) (Yang and Peng, 2013) are compared. Evaluation results are provided in Tables 4, 5 and 6. For TBHSA, the inertia weight ω is set to be 0.6 and parameters c_1 and c_2 are both set as 2.0. The number of maximum iteration G and the size of particle swarm *PNum* are set to be 200 and 50.

From Tables 4–6, it can be seen that the makespan and the total trust utility value generated by TSA and TBHSA are superior to those generated by SCPSA, but the cost generated by them are inferior to those generated by SCPSA. Because the trust constraints considered by them result in an extra cost. Besides, it is obvious that the schedule generated by TBHSA is better than that generated by TSA. The reason is that TBHSA uses a global search algorithm with strong search ability, whereas TSA uses a local search algorithm with weak search ability.

The proposed algorithm TBHSA is further compared with three representative metaheuristic algorithms: EA, ACO and PSO. Zhu and Wang (2008) constructed a novel task scheduling model focused on time and security constrains and proposed an EA to solve it. Chen and Zhang (2009) presented an ant colony optimization (ACO) algorithm based on seven new heuristics for large-scale workflows. In ACO, different QoS parameters, including reliability, time and cost, were considered to search a solution which could satisfy

Table 4
A schedule generated by the SCPSA.

Task	Resource set	$Time'(t_i)$	$Cost'(t_i)$	$TUtil(t_i)$
t_1	$\{cs_5, \{ss_6, ss_6, ss_6\}\}$	4.66	32	6.2
t_2	$\{cs_6, \{ss_5, ss_2, ss_6, ss_5\}\}$	204.4	56	8.5
t_3	$\{cs_5, \{ss_5, ss_2, ss_5\}\}$	224.6	38	7.1
t_4	$\{cs_3, \{ss_1, ss_2, ss_2, ss_5\}\}$	97.5	54	9.3
t_5	$\{cs_5, \{ss_6, ss_6\}\}$	38	22	4.25
t_6	$\{cs_6, \{ss_5, ss_5, ss_5, ss_4\}\}$	234.1	59	8.8
t_7	$\{cs_6, \{ss_5, ss_5, ss_3, ss_5\}\}$	246.4	67	8.1
t_8	$\{cs_5, \{ss_6, ss_6, ss_3, ss_6\}\}$	246.2	46	7.85
t_9	$\{cs_3, \{ss_2, ss_1, ss_2, ss_5\}\}$	248.6	53	9.15
t_{10}	$\{cs_6, \{ss_5, ss_3, ss_5\}\}$	270.1	51	5.5
$Makespan'(DAG) = 270.1$		$Cost'(DAG) = 478$		$TotTUtil(DAG) = 74.75$

Table 5
A schedule generated by the TBHSA.

Task	Resource set	$Time'(t_i)$	$Cost'(t_i)$	$TUtil(t_i)$
t_1	$\{cs_5, \{ss_6, ss_6, ss_6\}\}$	4.66	32	6.2
t_2	$\{cs_5, \{ss_6, ss_6, ss_6, ss_3\}\}$	158.7	64	9.6
t_3	$\{cs_6, \{ss_5, ss_2, ss_5\}\}$	91.9	38	7.1
t_4	$\{cs_3, \{ss_1, ss_2, ss_2, ss_5\}\}$	157.7	54	9.3
t_5	$\{cs_4, \{ss_3, ss_1\}\}$	42	19	5.75
t_6	$\{cs_6, \{ss_5, ss_5, ss_5, ss_4\}\}$	201.6	59	8.8
t_7	$\{cs_4, \{ss_1, ss_5, ss_3, ss_4\}\}$	172	64	9.45
t_8	$\{cs_6, \{ss_5, ss_3, ss_4, ss_2\}\}$	205.4	50	9
t_9	$\{cs_4, \{ss_2, ss_1, ss_4, ss_5\}\}$	189	49	9.3
t_{10}	$\{cs_6, \{ss_5, ss_3, ss_5\}\}$	221.5	51	5.5
$Makespan'(DAG) = 221.5$		$Cost'(DAG) = 480$		$TotTUtil(DAG) = 80$

Table 6
A schedule generated by the TSA.

Task	Resource set	$Time'(t_i)$	$Cost'(t_i)$	$TUtil(t_i)$
t_1	$\{cs_5, \{ss_6, ss_6, ss_6\}\}$	4.66	32	6.2
t_2	$\{cs_5, \{ss_6, ss_6, ss_6, ss_3\}\}$	158.7	64	9.6
t_3	$\{cs_6, \{ss_5, ss_2, ss_5\}\}$	91.9	38	7.1
t_4	$\{cs_3, \{ss_1, ss_2, ss_2, ss_5\}\}$	157.7	54	9.3
t_5	$\{cs_5, \{ss_6, ss_6\}\}$	164.2	22	4.25
t_6	$\{cs_6, \{ss_5, ss_5, ss_5, ss_4\}\}$	201.9	59	8.8
t_7	$\{cs_6, \{ss_5, ss_5, ss_3, ss_5\}\}$	214.06	67	8.1
t_8	$\{cs_6, \{ss_5, ss_3, ss_4, ss_2\}\}$	227.6	50	9
t_9	$\{cs_3, \{ss_2, ss_1, ss_2, ss_5\}\}$	200.7	53	9.15
t_{10}	$\{cs_6, \{ss_5, ss_3, ss_5\}\}$	236.3	51	5.5
$Makespan'(DAG) = 236.3$		$Cost'(DAG) = 490$		$TotTUtil(DAG) = 77$

all QoS constraints and optimize the QoS parameter according to the user's preference. Pandey *et al*. (2010) proposed a particle swarm optimization (PSO) algorithm for cloud workflow applications to minimize the total cost, including computation cost and data transmission cost. The following four scheduling problems are selected for testing: (T12, CS3, SS3), (T14, CS3, SS4), (T16, CS5, SS6), and (T20, CS6, SS6). In EA, the crossover rate and the mutation rate are respectively set to be 0.7 and 0.1. According to Wu *et al*. (2013), in ACO, the weights of heuristic information and pheromone are set as 2 and 1, respectively. The updating rate of local pheromone and global pheromone are all 0.1. The parameters in PSO are the same to those in TBHSA. For fairness, the fitness function, the maximum iteration and the number of new individuals of four algorithms are the same. The experimental results based on the three basic performance metrics of makespan, cost and the total trust utility value are provided in Table 7.

From Tables 7, we can draw the conclusion that the scheduling obtained from TBHSA is better than those obtained from EA, ACO, and PSO. TBHSA can map tasks to the trustful computation services for execution and retrieval of the required data sets from trustful storage services, thus improving the trustworthiness of execution environment.

Table 7
The results of performance indexes generated by four algorithms.

Performance	Instance	EA	ACO	PSO	TBHSA
$Makespan'(DAG)$	(T12, CS3, SS3)	442.8	403.2	383.6	330.5
	(T14, CS3, SS4)	565.3	522.4	502.1	480.7
	(T16, CS5, SS6)	743.2	696.8	613.5	570.2
	(T20, CS6, SS6)	871.2	810.9	783.7	732.5
$Cost'(DAG)$	(T12, CS3, SS3)	843.1	703.2	688.4	652.8
	(T14, CS3, SS4)	1091.3	978.7	893.1	870.5
	(T16, CS5, SS6)	1447.2	1306.3	1148.7	1090.1
	(T20, CS6, SS6)	1749.6	1502.5	1486.5	1330.6
$TotTUtil(DAG)$	(T12, CS3, SS3)	213.8	251.2	222.2	283.5
	(T14, CS3, SS4)	291.6	316.1	324.5	332.4
	(T16, CS5, SS6)	394.7	407.5	387.8	417.8
	(T20, CS6, SS6)	445.8	423.6	408.2	450.7

Table 8
The comprehensive utility values of (T50, CS8, SS10) generated by four algorithms.

	1	2	3	4	5	6	7	8
EA	0.913	0.861	0.863	0.889	0.895	0.892	0.913	0.923
ACO	0.965	0.912	0.926	0.928	0.913	0.925	0.969	0.965
PSO	0.920	0.929	0.912	0.921	0.965	0.971	0.958	0.982
TBHSA	0.976	0.969	0.986	0.993	0.962	0.975	0.988	0.992

Table 9
Significance test using Duncan for four algorithms.

Algorithm	Number	Subset for alpha = 0.05		
		1	2	3
EA	8	0.89363		
ACO	8		0.93788	
PSO	8		0.94475	
TBHSA	8			0.98013
Sig.		1.000	0.541	1.000

4.4.2. *Significance Test*

Furthermore, a significance test is used to evaluate the performances of four algorithms. Each experiment of the instance (T50, CS8, SS10) is repeated for 8 times with the same configuration parameters. According to Eq. (22), the comprehensive utility values of (T50, CS8, SS10) obtained respectively by four algorithms are provided in Table 8.

To compare the performances of four algorithms, according to Table 8, the significance test is carried out with SPSS (Statistical Product and Service Solutions) 19.0. One-way analysis of variance (ANOVA) is performed through post-hoc analysis with Duncan method. The significance level is set at 0.05. The results of significance test are shown in Table 9.

In Table 9, the first column lists the algorithms being tested, which are ordered from the smallest to the largest according to their average of 8 comprehensive utility values in

Table 8. The second column lists the sample number used to calculate their mean. The third column lists the comparison results at the significant level of 0.05 and the averages in the same child column indicate that the corresponding algorithm has no significant difference. For example, the averages (0.93788 and 0.94475) are in the same second child column, indicating that the corresponding algorithms of ACO and PSO have no significant difference. The last row of the table is the mean variance homogeneity test promotion rate, the value greater than 0.05 indicates that the variance among groups is homogeneous. According to Table 9, we can know that TBHSA shows the significant difference compared with EA, PSO, and ACO and that there is no significant difference between PSO and ACO, et al.

5. Conclusions and Future Work

By integrating the trust mechanism into the strategy for cloud workflow applications, the proposed trust-based scheduling algorithm TBHSA can make an effective allocation of tasks to trustful resource sets and improve the trustworthiness of execution environment. The main contribution of this paper includes formulating the trust relationship between tasks and resource sets and extending the traditional formulation of the scheduling problem. Experimental results indicate that TBHSA can get a higher comprehensive utility value than the traditional algorithms.

In the next phase of our research, we will study the way to describe the characteristics of trust and the trust relationships between resources sets and tasks more effectively.

Acknowledgement. This research is part of Projects 2009011022-2 and 2013011017-2 supported by Natural Science Foundation of Shanxi Province.

References

Abbadi, I.M., Ruan, A. (2013). Towards trustworthy resource scheduling in clouds. *IEEE Transactions on Information Forensics and Security*, 8(6), 973–984.

Abrishami, S., Naghibzadeh. M., Epema, D. (2013). Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Iteration Computer Systems*, 29(1), 158–169.

Al-Mistarihi, H.E., Yong, C. (2009) On fairness, optimizing storage service selection in data grids. *IEEE Transactions on Parallel and Distributed Systems*, 20(8), 1102–1111.

Bessis, N., Sotiriadis, S., Xhafa, F., Asimakopoulou, E. (2013). Cloud scheduling optimization: a reactive model to enable dynamic deployment of virtual machines instantiations. *Informatica*, 24(3), 357–380.

Chen, W.L., Wu, W.J., Shi, Y.H. (2010). A novel set-based particle swarm optimization method for discrete optimization problem. *IEEE Transactions on Evolutionary Computation*, 14(2), 278–300.

Chen, W.N., Zhang, J. (2009). An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Trans. on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 39(1), 29–43.

Choi, J., Choi, C., Yim, K., Kim, P. (2013). Intelligent reconfigurable method of cloud computing resources for multimedia data delivery. *Informatica*, 24(3), 381–394.

Cloudsim, The Clouds Lab (2012). *udsim: a framework for modeling and simulation of cloud computing infrastructures and services*Clo. Available at http://www.cloudbus.org/cloudsim/, accessed 28/02/2012.

Deelman, E. (2010). Grids and clouds: making workflow applications work in heterogeneous distributed environments. *International Journal of High Performance Computing Applications*, 24(3), 284–298.

Deelman, E., Gannon, D., Shields, M., et al. (2009). Workflows and e-science: an overview of workflow system features and capabilities. *Future Iteration Computer Systems*, 25(5), 528–540.

Fabio, D., Fagundez, A., Xavier, J. (2009). Continuous nonlinear programming techniques to solve scheduling problems. *Informatica*, 20(2), 203–216.

Foster, I., Zhao, Y., Raicu, I., et al. (2008). Cloud computing and grid computing 360-degree compared. In: *IEEE Grid Computing Environments*. IEEE Press, New York, pp. 60–69.

Garey, M.R., Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. In: *IEEE International Conference on Neural Networks*, pp. 1942–1948.

Kołodziej, J., Xhafa, F. (2011). Meeting security and user behavior requirements in grid scheduling. *Simulation Modelling Practice and Theory*, 19(1), 213–226.

Kołodziej, J., Xhafa, F. (2012). Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling. *Computers and Mathematics with Applications*, 63(2), 350-364.

Lin, S., Kernighan, B.W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2), 498–516.

Pandey, S., Wu L., Guru S., et al. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: *24th IEEE International Conference on Advanced Information Networking and Applications*. Institute of Electrical and Electronics Engineers Inc., Perth, pp. 400–407.

Rood, B., Lewis, M.J. (2008). Resource availability prediction for improved Grid scheduling. In: *Proc. of 4th IEEE Int. Conf. on eScience*. Inst. of Elec. and Elec. Eng. Computer Society, Indianapolis, pp. 711–718.

Song, S.S., Hwang, K., Zhou, R., et al. (2005). Trusted P2P transactions with fuzzy reputation aggregation. *IEEE Internet Computing*, 9(6), 24–34.

Song, S.S., Hwang, K., Kwok, Y.K. (2006). Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling. *IEEE Transactions on Computers*, 55(6), 703–719.

Venugopal, S., Buyya, R. (2008). An SCP-based heuristic approach for scheduling distributed cloud workflow application on global grids. J. Parallel Distrib. Comput, 68(4), 471–487.

Wang, M., Ramamohanarao, K., Chen, J. (2009). Trust-based robust scheduling and runtime adaptation of scientific workflow. *Concurrency and Computation: Practice and Experience*, 21(3), 1982–1998.

Wang, W., Zeng, G.S., Tang, D.Z., et al. (2012). Cloud-DLS: Dynamic trusted scheduling for cloud computing. *Expert Systems with Applications*, 39(3), 2321–2329.

Wu, Z.J., Lin X., Ni, Z.W., et al. (2013). A market-oriented hierarchical scheduling strategy in cloud workflow systems. *Journal of Supercomputing*, 63(1), 56–293.

Yang, Y.L., Peng, X.G. (2013). Trust-based scheduling strategy for workflow applications in cloud environment. In: *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. IEEE Press, pp. 316–320.

Zhu, H., Wang, Y.P. (2008). Security-driven task scheduling based on evolutionary algorithm. In: *Proc. of the Computational Intelligence and Security*. IEEE Press, New York, pp. 451–456.

Y.L. Yang received her MS degree in Computer Science and Technology from Guangxi Normal University, China, in 2007. Now she is studying for her PhD degree in College of Computer and Software at Taiyuan University of Technology, China. Her research interests are related with computer network security and cloud computing.

X.G. Peng graduated from Beijing Institute of Technology, China, in 2004 and received his PhD in Computer Science and Technology. He is now a professor in College of Computer science and technology, Taiyuan University of Technology, Taiyuan, China. His research interests mainly focus on computer network security, trusted computing and cloud computing.

J.F. Cao received her MS degree in Computer Science and Technology from Shanxi University, China, in 2005. Currently she is studying for her PhD degree in College of Computer and Software, Taiyuan University of Technology, China. Her research interests include neural networks, machine learning and affective computing.

Pasitikėjimu grindžiama planavimo strategija debesų darbo srautų taikomosioms programoms

Yuli YANG, Xinguang PENG, Jianfang CAO

Tradiciniai debesų darbo srautų taikomųjų programų planavimo tyrimai daugiausia yra orientuoti į kompromisą tarp užbaigimo laiko ir vykdymo išlaidų. Tačiau, saugumas ir patikimumas yra esminiai debesų darbo srauto planavimo veiksniai atviroje dinaminėje heterogeninėje debesų kompiuterijos aplinkoje. Šiame straipsnyje mes pristatome pasitikėjimu grindžiamą planavimo strategiją, remiantis visapusišku laiko, išlaidų, saugumo ir patikimumo įvertinimu. Pradžioje formuluojame debesų darbo srauto planavimo uždavinį su pasitikėjimo apribojimais ir pasiūlome naują pasitikėjimu grindžiamą debesų darbo srauto planavimo algoritmą, kuriame naudojamas aibėmis pagrįstas dalelių spiečiaus optimizavimo metodas, aibės padengimo uždavinys, medžio paieškos euristika. Eksperimento rezultatai rodo, kad lyginant su evoliuciniu algoritmu, skruzdžių kolonijos optimizavimu ir dalelių spiečiaus optimizavimu, pasiūlytas algoritmas yra efektingesnis.