EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Multi-heuristic scheduling methods for workflow in credit cloud

Zhang Xiaodong, Yao Yuan and Shen Hong*

*Correspondence:
shenhong@nau.edu.cn
School of Information and Engineering, Nanjing Audit University, Nanjing 211185, Jiangsu, China

Abstract

In the credit cloud, credit services are sold to applications for credit computing, credit fusion and credit risk estimates. Plenty of services with different performance for the same task may have different execution time and charged by various ways. The users have specific requirements for the workflow completion time or cost. Hence, to meet the user's satisfaction is an important challenge. In this paper, we propose heuristic scheduling methods for credit workflow with total cost minimization, and the deadline should be satisfied. The problem can be divided into two sub-problems, task-mode mapping and task tabling on renting service instances. For the task-mode mapping problem, a recursive heuristic method is constructed to select appropriate service for each task of the workflow. Then another heuristic algorithm based is established to get a final schema with deadline constraint. We discussed the service instance rented in shareable manner and compared with un-shareable manner. Three renting strategies are discussed in detail. Experimental results show the effectiveness and efficiency of the proposed algorithm.

Keywords: Credit services, Heuristic scheduling methods, Renting manner

1 Introduction

Credit information is everywhere, such as finance, medical care, e-purchase platforms and governments. Distributed credit information service platform is urgently needed to realize the integration of public credit services. The final platform is based on the credit cloud computing for rapid credit computing, credit transferring and data sharing. Users can obtain the credit information via credit cloud under certain terms. The large-scale credit information applications expressed through workflow can be responded by multi heterogeneous platforms services via the credit cloud.

Workflows are always denoted by directed acyclic graphs (DAG), in which nodes represent tasks and arcs represent precedence relations between tasks. The essence of workflow scheduling problem is to map the tasks to resources with a balance between the task execution time and resource costs. This problem has been proved to be NP-hard [1, 2]. In recent years, the cloud workflow scheduling problem has been widely concerned by many researchers [3–7]. Many existing works about deadline constrained cloud workflow scheduling problem are focused on objectives such as minimization of cost and maximization of reliability and utilization [8], and multi-objectives [9]. Usually,

the workflow applications can be executed on cloud platforms, such as Google cloud and Amazon EC2.

In the cloud environment, the IaaS center usually provides three resource renting manners to users, reserved, on-demand and spot [10]. Many types of virtual machines provide different levels of services with various prices. Usually, resource with high computing and storage performance has higher rental costs; thus, the shorter the execution time, the higher the service costs [11]. And the cloud services are usually priced in intervals, such as Amazon EC2, IBM cloud. In such cases, the free slots of rented service instances can be shared among the tasks of the same workflow to decrease the total renting cost. In PaaS or SaaS platforms, different kinds of platforms and software are encapsulated into services and sold to customers. All of these resources provided by clouds are called cloud services. Once the cloud services are charged by intervals and can be shared among tasks are called shareable services, and otherwise, they are termed as unshareable services. Many recent works study on un-shareable service selection for workflow tasks [10, 12, 13]. In these papers, heuristic and meta-heuristic methods are used to select services for the workflow tasks with cost minimization. However, to the users, they care more about the total cost of the service rented since the resource is concerned to be unlimited in the cloud environment. When services are priced in interval based models [14–16], the free time of the rented services can be shared among the tasks to reduce the renting cost. Elastic resource provisioning is discussed based on the interval-based charging models, and more realistic factors, such as VM setup time and data transferring time, are concerned. Heuristic method is proposed to solve this practical problem [17–19].

In the cloud service environment, such as vehicle network cloud or Internet of things cloud, some scholars have studied the optimization of resource services such as network transmission performance [20], network rate [21], transmission efficiency [22, 23], optimization of joint pricing and power allocation [24] and achieved some results. These studies provide a technical reference for the transmission of credit cloud services on distributed heterogeneous platforms. The credit services are provided in PaaS platform for users to rent. Usually, the credit services provide two renting manners, single renting which is charged on demand according to the length of the lease, and entire renting which is charged by quantity or reserved time intervals with certain discount. With the explosion of credit information and the growth of credit applications, the cloud credit service pricing strategies become more and more worth discussing. Consider the workflow under credit cloud, an efficient scheduling method is necessary for higher resource utilization and cheaper renting cost based on shareable service instances.

In this paper, we discuss the credit service workflow scheduling problems, where users can rent scalable credit services, and the goal is cost minimization of the rented services. The credit services can be obtained on-demand with different costs. Service instances can be shared among the tasks to reduce cost. That is the rented time intervals can be utilized by other tasks, which choose the same service. Finally, the total cost is minimized and the deadline should be satisfied. A recursive method based on the workflow structure is proposed to solve the task-mode mapping problem, which is used to select appropriate service for each task in the workflow. Then an optimal mode assignment schema is obtained. Then a problem-specific heuristic algorithm is constructed to

table the tasks to proper time slot of the existing service instances to reduce the cost, and the deadline should be satisfied. In the experimental tests, three renting manners are discussed in detail, single renting manner, entire renting manner, single and entire renting hybrid manner. Different from the two renting manners mentioned before, the hybrid manner considers the services can be rented by the combination of on-demand and reserved time-intervals. The service shareable renting mode is compared with the un-shareable mode. The results show the effectiveness and efficiency of the proposed algorithm.

2 Problem description

2.1 User workflow applications

A workflow application can be described by an Task-on-Node (AON) directed graph $G=(V, A)$, where V is the set of task nodes $\{v_o, v_1, v_2, \ldots, v_n, v_{n+1}\}$ and arc set A represents the finish-start zero-lag precedence relations. Any $(v_i, v_j) \in A$ indicates task v_i be the immediate predecessor of task v_j; in other words, task v_j cannot start before task v_i has finished. Task v_0 and task v_{n+1} are dummy tasks, representing the starting and the finishing of the application, respectively. Let $Dpre_j$, $Dsucc_j$ denote the set of immediate predecessors, immediate successors of task j, respectively. $|Dpree_j|$ and $|Dsucc_j|$ represent the number of immediate predecessors and successors, respectively. Let st_j be the starting time of task j, and ft_j be the finishing time of task v_j. The deadline of the workflow is δ, that is the finishing time of task v_{n+1} should satisfy $ft_{n+1} \leq \delta$.

2.2 Cloud service

Various types of services (such as credit computing, credit transferring, credit estimation, crediting data, infrastructures and software) are provided by the credit cloud providers. The workflow should rent proper service instances to execute the tasks. Let $S = (S^1, S^2, \cdots, S^w)$ be the service set of the system. Each type of the services is charged by time intervals, single renting, entire renting or hybrid renting. The on-demand single renting price per unit $price^i$, entire renting time interval $Rinterval^i$, and the discount $disc^i$ of the entire renting of each service are already known. Users send the workflow applications to the cloud center; the Workflow Scheduler should adopt efficient scheduling algorithms to map different services to the tasks with distinct execution time and cost.

There are several candidate services which can perform task v_j. That is, each task v_j has several execution modes. Let M_j be the candidate service mode set of v_j, $M_j = \{M_j^1, M_j^2, \cdots, M_j^k\}$. The mode M_j^k is a two-tuples (S^k, d_j^k), where S^k means the selected service, and d_j^k is the execution time of the kth mode of v_j. The cost of the current mode c_j^k can be obtained by $d_j^k \times price^k$.

2.3 A workflow example

A workflow example is demonstrated in Fig. 1. The candidate modes are shown beside the tasks. The price list of the cloud services of the system is presented in Table 1. Three renting manners, single renting, entire renting and hybrid renting, are illustrated in detail.

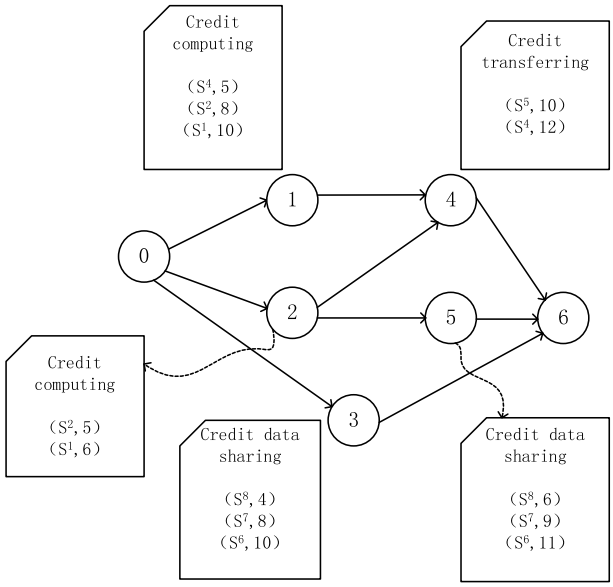

Fig. 1 An example of the workflow

Table 1 The service price model

Services	**Cost(hour)** $price^i$	**Interval(hours)** $Rinterval^i$	**Discard** $disc^i$
S1	3	10	0.8
S2	4	12	0.9
S3	6	10	0.9
S4	7	24	0.7
S5	9	24	0.7
S6	9	8	0.95
S7	12	8	0.9
S8	25	10	0.8

There are three execution modes for v_1 to carry out credit computing. If the on-demand single renting manner is adopted, the execution time and cost of v_1 is as follows. In the first mode, service S^4 is selected. The execution time is 5, and it costs 35. The second mode needs 8 time units with service S^2, and it costs 32. And the last mode needs 10 time units, it costs 30 for service S^1. Undoubtedly, the service S^4 provides the best computing power with higher renting cost. If the entire renting manner is selected, it just needs only one lease of service S^1 since $\lceil 10/10 \rceil = 1$, and the cost of v_1 is 24. Considering task v_5 and service S^6 is chosen, it costs $9 \times 11 = 99$ on single renting manner for 11 time units, while it should pay $\lceil 11/8 \rceil \times 8 \times 9 \times 0.95 = 136.8$ for the entire renting manner. However, if the hybrid renting manner is adopted, it just need $\lfloor 11/8 \rfloor \times 8 \times 9 \times 0.95 + 11\%8 \times 9 = 95.4$. Compared with single renting, the reduction of the cost is 3.64%. The cost has been reduced by 30.3% comparing with entire renting. Hence, it is necessary to choose proper renting manner for each task in order to reduce the total cost of the workflow. Furthermore, if the rented service instance can be shared among the tasks which select the same service, the free slots

of the time intervals can be utilized to reduce the total renting cost of the workflow. The scheduling strategy under sharable service instances will be discussed in detail in Sect. 3.

2.4 The considered scheduling problem

Given the deadline of the workflow δ, the service set of the system $S = (S^1, S^2, \cdots, S^w)$ and the their pricing model, the objective of the workflow scheduling algorithm is to select appropriate number and type of service instances making a balance between execution time and renting cost. The considered problem mainly contains two sub-problems: mode assignment and task sequencing. The mode assignment tries to assign an appropriate mode to each task. The task sequencing tries to obtain an optimal schema to minimize the total renting cost, while the service instances can be shared by the tasks which choose the same service.

3 Proposed method

3.1 A heuristic method for service selection of the workflow

As mentioned above, the first sub-problem of the considered workflow scheduling problem is mode assignment. Each task has several selective services with different execution time and cost. For a workflow with $n+2$ tasks, there may be $\prod_{i=0}^{n} |\mathrm{M}_i|$ service mode selection schemas. With a given deadline, some of them are feasible solutions, and the others are unfeasible solutions. We propose a heuristic method for obtaining an optical solution $\mathrm{Mode} = (M_1^*, M_2^*, \cdots, M_n^*)$ satisfying the deadline with less cost. The main idea of the heuristic method is described as follows. Firstly, select the service with shortest execution time for each task. Obviously, the mode selection schema can meet the deadline, and the resulting cost is most expensive. Then, the schema is optimized recursively by adjusting the service mode of some tasks with less cost, while the deadline is still satisfied.

In the service pool of each task, the service mode of each task is ranked in ascending order of the execution time, and the descending order of the cost. That is, $d_i^{k+1} > d_i^k$ and $c_i^{k+1} < c_i^k, 0 \le k \le |\mathrm{M}_i| - 1$. For a give mode selection schema, the kth mode is selected for task i, the earliest start time and finish time can be obtained by $est_i = \max\limits_{j \in Dpre_i} \{eft_j\}$, $eft_i = est_i + d_i^k$. With the given deadline δ, the latest start time and finish time is obtained by $lft_i = \min\limits_{j \in Dsucc_i} \{lst_j\}$, $lst_i = lft_i - d_i^k$. If $lft_i - est_i \ge d_i^{k+1}$, a service mode M_i^{k+1} can be reselected for task i for cheaper cost. There may be several tasks can be adjusted their service modes in set Ψ. In each recursive, the most appropriate task is selected to choice a cheaper service, which is defined by the reduction of cost in per unit of time $\tau_i = (c_i^k - c_i^{k+1})/(d_i^{k+1} - d_i^k)$. Furthermore, the degree centrality of the node in the workflow is considered, since the smaller the degree of the selected task, the smaller of the influence on other nodes. $\tau_i' = \tau_i/(|Dsucc_i| + |Dpre_i|)$ is computed for each task $i \in \Psi$. Algorithm is described below.

Algorithm 1 A recursive heuristic method for mode assignment

1) $\Psi = \phi, k = 0, \text{Mode} = (M_1^k, M_2^k, \Lambda, M_n^k)$
2) $est_0 = 0,\ eft_0 = 0,\ lst_{n+1} = \delta_{n+1},\ lft_{n+1} = \delta_{n+1}$
3) For $i = 0$ to $n+1$
4) $\quad est_i = \max_{j \in Dpre_i} \{eft_j\},\ eft_i = est_i + d_i^k$
5) Endfor
6) For $i = n+1$ to 0
7) $\quad lft_i = \min_{j \in Dsucc_i} \{lst_j\},\ lst_i = lft_i - d_i^k$
8) Endfor
9) For $i = 0$ to $n+1$
10) If $k \le |M_i| - 1$ and $lft_i - est_i \ge d_i^{k+1}$
11) $\quad \Psi \leftarrow \Psi \cup \{i\}$
12) Endif
13) Endfor
14) While $\Psi \ne \phi$
15) For $\forall i \in \Psi$
16) $\quad \tau_i = (c_i^k - c_i^{k+1})/(d_i^{k+1} - d_i^k)$
17) $\quad \tau_i' = \tau_i / (|Dsucc_i| + |Dpre_i|)$
18) Endfor
19) $a \leftarrow \arg\max_{\forall j \in \Psi} \{\tau_j'\},\ \text{Mode} = (M_1^k, \Lambda, M_a^{k+1}, \Lambda, M_n^k)$
20) Endwhile
21) Return Mode

3.2 An example of the scheduling strategies

Traditional service scheduling problems discuss the service renting in non-shareable manner. However, the cloud services mostly can be shared between the tasks during the renting interval to decreased the total renting cost, since Minimizing service renting cost is more important to the cloud users rather than the system performance of service providers.

As described in the second section, if we can schedule the tasks who choose the same service to the same service instance, and entire rent the service with time intervals, the renting cost can get a discount. However, it is time consuming to test every time point to confirm whether there are two tasks can share the renting interval. Since the cloud service is thought to be unlimited and available anytime, the workflow can be run when submitted. The earliest start time of each task is obtained based on the structure through forward scheduling. Then, we will give the renting strategy to schedule the tasks efficiently.

A workflow with 12 tasks is shown in Fig. 2, and the deadline is set to 30. Based on the forwarding scheduling, the schema of the workflow is obtained as Fig. 5, where the service instances are un-shareable. To improve the scheduling efficiency, the renting

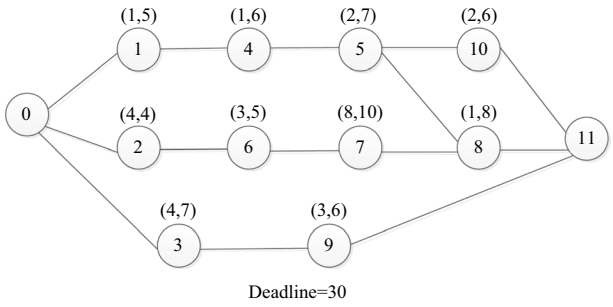

Fig. 2 An example of the problem model

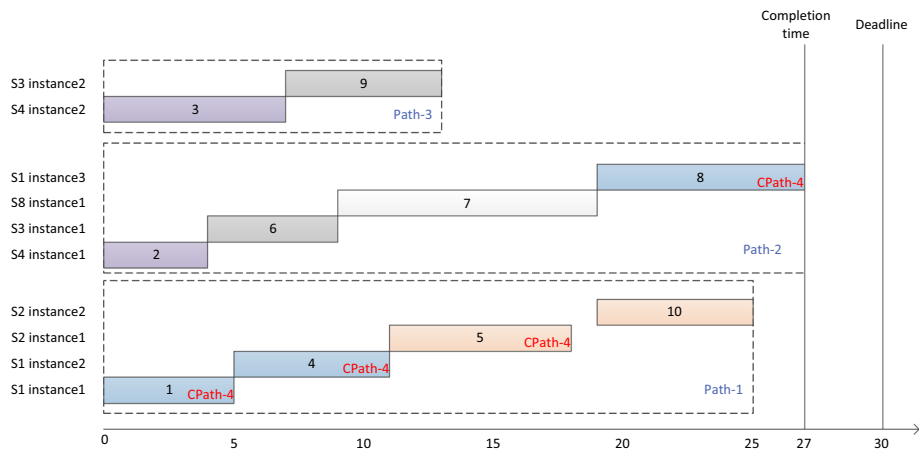

Fig. 3 A schema under un-shareable services

starting point and finishing point are decided whether the service instance can be shared among the tasks.

The considered workflow has four paths as shown in Fig. 3, and the critical path is 1-4-5-8. The completion time of the workflow is 27. It is finished 3 units before the deadline. That is, the workflow has 3 slack time units. So we can delay the tasks in path-3 (3,9) to share the existing Service4 instance1. Task 5 also can be delayed to share the renting interval of Service2. Task 9 should be retabled to share the existing Service3 instance1. It is sure that once two tasks on the same path selected the same service, they should rent the service in time interval manner, such as task 1 and 4.

The final schedule of the workflow is shown in Fig. 4. As we can see tasks numbered in 3,5,6,7,8,9 are delayed to start, but the completion time of the workflow still satisfies the deadline. 6 instances should be rented to execute the workflow. It is necessary to mention that entire rent the server instance in time intervals or on-demand single rent should be decided which way is cost-efficient. For example, task v_8 need

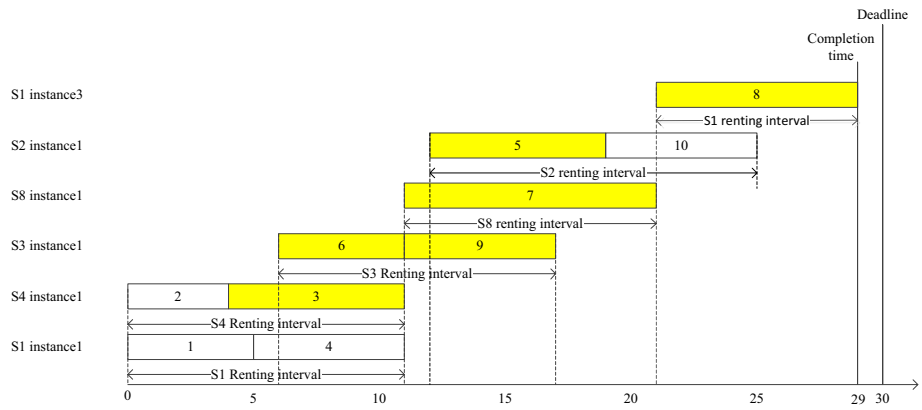

Fig. 4 A optimized schedule on shareable services

8 time units to finish by S^1, renting in per unit charged manner should be chosen for energy conservation. Task v_5 rents S^2 instance1 in one interval and shared 5 units slot to v_{10}. In this case, task v_{10} still need 1 unit time. Then the renting method mentioned above is adopted to reduce the cost.

3.3 Heuristic scheduling method proposed for optimizing the cost under shareable instances

Given the deadline δ, the selected modes for each task $\text{Mode} = (M_1^*, M_2^*, \cdots, M_n^*)$ obtained by HSI, the proposed heuristic scheduling method is used to schedule the tasks sequentially satisficing the precedence constraints, and the services can be shared on the same time interval to reduce the cost.

According to the mode assignment schema, each task has selected a service, and the execution time is fixed. The earliest start time est_i and finish time eft_i of the task i is computed by the forwarding scheduling, and the latest start time lst_i and finish time lft_i is obtained by the backward scheduling. For each type of service k in the service set $S = (S^1, S^2, \cdots, S^w)$, we construct a set of tasks which has select service k, and the tasks are ordered in the non-descending earliest start time, $\Re_k = \{a_1, a_2, \cdots, a_m\}$. The proposed heuristic scheduling method is tried to reschedule the tasks in $\Re_k$ to appropriate start time sharing the renting interval of the service to reduce the total cost.

In order to describe our heuristic scheduling method under shareable instances in detail, we give a mathematical description of some variables. First, the number of existing instances of service S^k is defined as $NIns_k$. Then, we define two sets, $TInterval_k^t$ and $TSlot_k^t$, to record the renting time intervals and the remaining time slots of the existing instance tth of service S^k. The rth renting interval in set $TInterval_k^t$ can be described as $[st_r^T, ft_r^T]$, where st_r^{TI} and ft_r^{TI} separately represent the starting time and finishing time of the renting interval. The sth slot in set $TSlot_k^t$ can be described as $[st_s, ft_s]$, where st_s^{TS} and ft_s^{TS} separately represent the starting time and finishing time of the slot. The tasks in $\Re_k$ tries to schedule in optimal start time sharing the existing service instance. The task will rent a new service instance when all the existing remaining time intervals cannot satisfy its time constraints.

Algorithm2 Heuristic Scheduling method under Shareable Instances

1) For $k = w$ to 1
2) Sequence the tasks which select service S^k by non-descending earliest start time in $\Re_k$
3) Set $t = 1, TInterval_k^t = \phi$, $TSlot_k^t = \phi$, $NIns_k = 1$
4) Scheduling the first task a_1 in $\Re_k$ in $[est_{a_1}, eft_{a_1}]$
5) Update $TInterval_k^t = TInterval_k^t \cup \{[est_{a_1}, eft_{a_1}]\}$
6) Update $TSlot_k^t = \{[eft_{a_1}, +\infty]\}$
7) For $l = 2$ to $|\Re_k|$
8) $t = 1$
9) While $t \le NIns_k$
10) If find a slot $s : [st_s^{TS}, ft_s^{TS}] \in TSlot_k^t$ and satisfy $est_{a_l} = st_s^{TS}$, $eft_{a_l} <= ft_s^{TS}$
11) Schedule a_l between est_{a_l} and eft_{a_l}
12) Update $ft_s^{TI} = eft_{a_l}$, $st_s^{TS} = eft_{a_l}$
13) Break
14) Endif
15) If find a slot $s : [st_s^{TS}, ft_s^{TS}] \in TSlot_k^t$ and satisfy $est_{a_l} < st_s^{TS}$, $lst_{a_l} >= st_s^{TS}$, $eft_{a_l} <= ft_s^{TS}$
16) Schedule task a_l between st_s^{TS} and $st_s^{TS} + d_{a_l}^*$
17) Update $ft_s^{TI} = st_s^{TS} + d_{a_l}^*$, $st_s^{TS} = ft_s^{TI}$
18) Update the earliest start time of all successor tasks of a_l
19) Update the latest finish time of all predecessor tasks of a_l
20) Break
21) Endif
22) If $t = NIns_k$
23) Schedule a_l between est_{a_l} and eft_{a_l} with a new service instance
24) Update $TInterval_k^{t+1} = \phi \cup \{[est_{a_l}, eft_{a_l}]\}$, $TSlot_k^{t+1} = \phi \cup \{[eft_{a_l}, +\infty]\}$
25) $NIns_k = NIns_k + 1$
26) Break
27) Endif
28) $t = t + 1$
29) Endwhile
30) Endfor
31) Endfor

The final total cost of the workflow is computed based on the service instance renting time intervals recorded in $TInterval_k^{ins}$, and the hybrid renting manner is adopted.

4 Experimental and results

The above algorithms were coded in Java and plenty of workflow instances with various different size were executed on a personal computer with Core(TM) i5-5200U CPU 2.2 GHz 2.19 GHz, 8 GB RAM on the MS Windows 10 operating system. The standard test sets 30, 60, 90, 120 from PSPLIB (http://129.187.106.231/psplib) have been adopted here to evaluate the algorithms. Random workflow instances with $|V| \in \{150, 210, 270\}$ are generated by [25]. The topology complexity of the workflow is measured by OS (order strength), and OS is set to 0.1 in this paper.

4.1 Parameter setting

In our experiments, the total number of service type is set to 10. The price of service w in per unit is generated by convex function, and the service set $S = (S^1, S^2, \cdots, S^{10})$ of the system is sorted in ascending order of the price. The entire renting time period for each service is set to 10 units, and the discount of this renting mode is set to 0.8. Each

task of the workflow can rent the service in three manners: single renting charged by on-demand units, entire renting charged by time intervals with discount, hybrid renting of the above two modes.

The candidate service mode M_i of task i is uniformly distributed in the interval $[2, 10]$ as described in [26]. M_i services are selected sequentially from $S = (S^1, S^2, \cdots, S^{10})$. The duration d_i^k of the current selected service k is generated randomly by [26] in the interval $[3, 120]$, $d_i^{k+1} > d_i^k$. The resulting on-demand cost c_i^k satisfies $c_i^{k+1} > c_i^k$.

The 10 different deadlines for each workflow is constructed to further compare the algorithms. Deadline δ_{n+1} is defined as $\delta_{n+1} = Minest_{n+1} + 0.05 \times DF \times (Maxest_{n+1} - Minest_{n+1})$, deadline factor $DF \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $Minest_{n+1}$ is the critical path of the current workflow in which the tasks select the service mode with shortest execution time, and $Maxest_{n+1}$ is the critical path of the workflow where the tasks select the cheapest service mode.

4.2 Computational results

In the following experiments, the service mode assignment schema is optimized by the first algorithm HIS proposed in this paper. Based on the resulting $mode = (m_0^*, m_1^*, \cdots, m_{n+1}^*)$, the critical path length the workflow is obtained by. Then, *Scost* defines the total cost by the single renting manner under the optimal mode assignment schema. Also, the total cost *Icost* is computed with the entire renting manner. The cost of the hybrid renting manner is recorded in *Hcost*. It is important to mention that in the above three workflow scheduling manners, the service instances are un-shareable among the tasks. The shareable service instances renting time intervals are obtained by the second proposed algorithm HSSI, and the optimal cost *Ocost* is computed by the hybrid renting manner.

The results on 7 different scale workflows are shown in Table 2. 10 different deadlines are performed on each workflow, and the average costs of *Scost*, *Icost*, *Hcost*, *Ocost* are compared. Since entire renting manner can get a discard of service, *Icost* is superior to *Scost* in general. The hybrid manner combines time interval renting and single renting based on the execution time can reduce the total cost more. The optimized *Ocost* under shareable service instance has better performance on all test cases. Specifically, on the workflow case with 30 tasks, the average *Scost* is 7626, and the average *Ocost* is 6218. On the workflow case with 270 tasks, the average *Scost* is 47198, and the average *Ocost* is 38397.

Table 2 The results on different problem sizes

Problem size	*S* cost	*I* cost	*H* cost	*O* cost
30	7626	6983	6247	6218
60	15,849	14,764	13,196	13,024
90	18,805	17,478	15,652	15,431
120	25,944	23,663	21,495	21,214
150	33,545	30,601	27,533	27,342
210	31,592	29,311	26,230	25,610
270	47,198	43,512	39,139	38,397

Table3 The performance of our algorithm

Problem size	ARPD (%)			
	Scost	Icost	Hcost	OCost
30	22.64	12.31	0.48	-
60	21.69	13.36	1.32	-
90	21.86	13.26	1.43	-
120	22.30	11.54	1.32	-
150	22.68	11.92	0.70	-
210	23.35	14.45	2.42	-
270	22.92	13.32	1.93	-

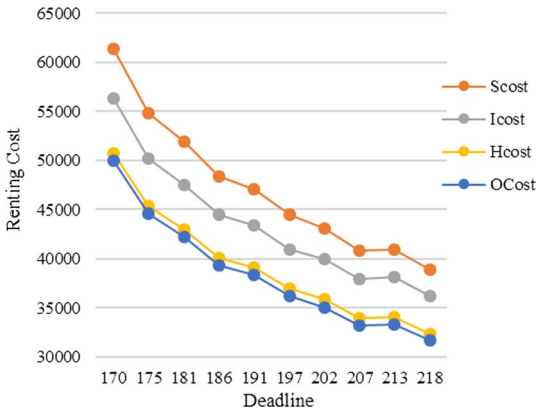

Fig. 5 The deadline influence on the workflow

We define $ARPD = (\text{Cost}_A - \text{Cost}_{Opt})/\text{Cost}_{Opt} \times 100\%$ to measure the performance of the algorithm. Cost_A represents the total cost of the workflow under the current renting manner. Cost_{Opt} is the optimal solution among the four renting manner. The results are shown in Table 3. As we can see, the maximal deviation from our proposed algorithm is 23.35% on the workflow with 210 tasks, and the average deviation of *Hcost* is 1.37%. The proposed algorithm can always obtain the optimal cost.

Finally, the deadline influence on the workflow is shown in Fig. 5. With the increase of deadline factor, the better the performance of our algorithm. The reason is that a larger deadline results in more candidate cheaper services for the tasks, and the total cost is decreased. Our algorithm can get better performance.

5 Discussion

In this paper, we propose two heuristic algorithms to minimize the renting cost of the credit cloud service workflows meeting the deadline of the user's. Algorithm 1 is constructed to obtain an optimal mode assignment schema for the workflow. Based on the optimal mode assignment schema, Algorithm 2 tries to schedule the tasks sharing the existing renting time intervals among the tasks which choice the same service, and reduces the total cost. Experimental tests are compared on the three renting manner, single renting, entire renting, and hybrid renting under the un-shareable service

instances, which is adopted in most of the existing researches. On the contrary, our algorithm takes advantage of the shareable service renting intervals, and computes the total renting cost in hybrid manner. The results show our method is efficient on solving these problems.

Abbreviations
DAG: Directed Acyclic Graphs; AON: Task-on-Node; HIS: Heuristic method for initial solution; HSSI: Heuristic Scheduling method under Shareable Instances.

Acknowledgements
The authors would like to thank Jiangsu Key Laboratory of audit information engineering for their support and anyone who supported the publication of this paper.

Funding
This work is supported by National Key R&D Program of China(Grant No. 2019YFB1404602).

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Ethics approval and consent to participate
Ethical approval.

Consent for publication
Not applicable.

Competing interests
The author declares that this article has no competing interests.

Received: 10 October 2021 Accepted: 23 November 2021
Published online: 07 December 2021

References

1. P. De, E. Dunne, J. Ghost et al., Complexity of the discrete time-cost tradeoff problem for project networks. Oper. Res. **45**(2), 302–306 (1997)
2. E. Demeulemeester, B. De Reyck, B. Foubert et al., New computational results on the discrete time/cost trade-off problem in project networks. J. Oper. Res. Soc. **49**(11), 1153–1163 (1998)
3. S. Abrishami, M. Naghibzadeh, D. Epema, Cost-driven scheduling of grid workflows using partial critical paths. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1400–1414 (2012)
4. S. Abrishami, M. Naghibzadeh, D.H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener. Comput. Syst. **29**(1), 158–169 (2013)
5. Z. Cai, X. Li, J. Gupta, Critical path-based iterative heuristic for workflow scheduling in utility and cloud computing, in *International Conference on Service-Oriented Computing: Service-Oriented Computing, 2013*, pp. 207–221.
6. L. Chen, X. Li, Cloud workflow scheduling with hybrid resource provisioning. J. Supercomput. **74**, 6529–6553 (2018)
7. Y. Yuan, X. Li, Q. Wang et al., Deadline division-based heuristic for cost optimization in workflow scheduling. Inf. Sci. **179**(15), 2562–2575 (2009)
8. G. Ismayilov, H.R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Futur. Gener. Comput. Syst. **102**, 307–322 (2020)
9. S. Saeedi, R. Khorsand, S. Bidgolt et al., Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. Comput. Ind. Eng. (2020). https://doi.org/10.1016/j.cie.2020.106649
10. J. Yu, R. Buyya, C. Tham, Cost-based scheduling of scientific workflow applications on utility grids, in *Processing of the 1st International Conference on E-Science Grid Computing*, 2005, p. 8.
11. Z. Cai, X. Li, J. Gupta, Heuristics for provisioning services to workflows in XaaS clouds. IEEE Trans. Serv. Comput. **9**(2), 250–263 (2016)
12. V. Ruben, K. Vanmechelen, J. Broeckhove, Cost optimal scheduling in hybrid IaaS clouds for deadline co constrained workloads, in *Proceedings of IEEE 3rd International Conference on Cloud Computing, 2010*, pp 228–235.
13. J. Durillo, R. Prodan, Multi-objective workflow scheduling in amazon EC2. Cluster Comput. **17**(2), 169–189 (2014)
14. AmazonEC2 (2016) Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/pricing.
15. E.K. Byun, Y.S. Kee, J.S. Kim et al., BTS: Resource capacity estimate for time-targeted science workflows. J. Parallel Distrib. Comput. **71**(6), 848–862 (2011)
16. X. Li, Z. Cai, Elastic Resource provisioning for cloud workflow applications. IEEE Trans. Autom. Sci. Eng. **14**(2), 1–16 (2017)
17. J. Durillo, R. Prodan, Multi-objective workflow scheduling in amazon EC2. Clust. Comput. **17**(2), 169–189 (2014)
18. N. Rizvi, R. Dharavath, D.R. Edla, Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization. Simul Model Pract Theory **110**(3), 102328 (2021)

19. A. Iranmanesh, H. Naji, DCHG-TS: A deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Cluster Comput. **24**(6), (2020)
20. X. Liu, X. Zhang, NOMA-based resource allocation for cluster-based cognitive Industrial Internet of Things. IEEE Trans. Ind. Inf. **16**(8), 5379–5388 (2020)
21. X. Liu, X. Zhai, W. Lu et al., QoS-guarantee resource allocation for multibeam satellite Industrial Internet of Things with NOMA. IEEE Trans. Ind. Inf. **PP**(99), 1–1 (2019)
22. L. Xin, X. Zhang, Rate and energy efficiency improvements for 5G-based IoT with simultaneous transfer. IEEE Trans. Ind. Inf. **PP**(99), 1–1 (2018)
23. X. Liu, X. Zhang, M. Jia et al., 5G-based green broadband communication system design with simultaneous wireless information and power transfer. Phys. Commun. **28**, 130–137 (2018)
24. W. Yuan, Tengyi, et al., Joint pricing and power allocation for dynamic spectrum access networks with stackelberg game model. IEEE Trans. Wirel. Commun. **10**(1) (2011)
25. E. Demelemeester, M. Vanhoucke, W. Herroelen, Rangen: a random network generator for activity-on-the-node networks. J. Sched. **6**(1), 17–38 (2003)
26. C. Akkan, A. Drexl, A. Kimms, Network decomposition based benchmark results for the discrete time-cost tradeoff problem. Eur. J. Oper. Res. **165**(2), 339–358 (2005)

Publisher's Note