A Reinforcement Learning Based Workflow Application Scheduling Approach in Dynamic Cloud Environment

Yi Wei[1(✉)], Daniel Kudenko[2,3], Shijun Liu[1], Li Pan[1], Lei Wu[1], and Xiangxu Meng[1]

[1] School of Computer Science and Technology, Shandong University, Jinan, China
weiyi9008@hotmail.com, {lsj,panli,i_lily,mxx}@sdu.edu.cn

[2] Department of Computer Science, University of York, York, UK
daniel.kudenko@york.ac.uk

[3] Saint Petersburg National Research Academic University of the Russian Academy of Sciences, St Petersburg, Russia

Abstract. Workflow technology is an efficient means for constructing complex applications which involve multiple applications with different functions. In recent years, with the rapid development of cloud computing, deploying such workflow applications in cloud environment is becoming increasingly popular in many fields, such as scientific computing, big data analysis, collaborative design and manufacturing. In this context, how to schedule cloud-based workflow applications using heterogeneous and changing cloud resources is a formidable challenge. In this paper, we regard the service composition problem as a sequential decision making process and solve it by means of reinforcement learning. The experimental results demonstrate that our approach can find near-optimal solutions through continuous learning in the dynamic cloud market.

Keywords: Cloud computing · Infrastructure as a service
Service composition · Markov decision process · Q-learning

1 Introduction

With the rapid development of cloud computing technologies, a large number of cloud-based applications are delivered across the internet. Cloud workflow applications are typical ones which can be used to handle complex and combinatorial tasks in academia and industry. For example, Cloud-based design and manufacturing (CBDM) [1] is a popular collaborative working pattern which utilizes cloud resources to execute applications to achieve cooperation across multiple enterprises. Scheduling workflow applications in clouds means using compositional cloud services to execute tasks contained in workflows. In the cloud market, IaaS providers provides various on-demand computing resources in the form of virtual machines (VMs). SaaS providers can use them to run their applications cost-effectively. However, how to choose appropriate VM instances to execute workflow tasks is not easy. Relationships among tasks need to be considered and there are multiple alternative VM instances with various configurations and prices provided by different IaaS providers.

Several works have been proposed to address workflow scheduling problems in the literature [2, 3]. Most studies are based on the assumption that the execution time of each task on different types of VM instances has been known in advance and it will not change. However, this assumption is not reasonable because SaaS providers hardly know the real performance of VMs in a dynamic cloud environment. In addition, some factors such as task parallelization and data transmission are seldom considered. Our main contribution of this paper is taking into account all these features in our model and proposing an effective workflow scheduling approach based on reinforcement learning. Compared with existing methods, our approach can learn the dynamic performance of the cloud market and generate near-optimal workflow application scheduling without any prior knowledge.

The rest of this paper is structured as follows. Section 2 gives an overview of relevant related work. In Sect. 3, we describe the three-tier cloud market model and then present the problem formulation in this context. Section 4 introduces the basic theories of reinforcement learning and details our approach. Experimental results and conclusion are given in Sects. 5 and 6 respectively.

2 Related Work

In recent years, the research on scheduling cloud based workflow applications has been studied intensively [2, 3]. Generally, monetary cost and makespan are two optimization objectives in this field. In our work, we focus on minimizing the workflow cost to help SaaS providers get more profits. There have been some existing approaches [4, 5] regarding this topic, but their considerations of workflow constraints and operating environment are inadequate. Most of them only consider some basic constraints (e.g. budget and deadline) and ignore the uncertain and dynamic characteristics of cloud markets.

In contrast, selecting services for workflow scheduling in the unknown and changing cloud environment can be viewed as Markov Decision Processes (MDPs), which can be solved by reinforcement learning approaches naturally. Some researches [6, 7] have applied Q-learning algorithm to achieve service composition. However, their problem models are very general and of course does not take into account the features of workflow applications. Similar to our objective, [8, 9] try to help SaaS providers to optimize resource allocation to their applications. But their application model is not a workflow thus it is different from our workload-based model.

3 Problem Statement

3.1 System Model

As shown in Fig. 1, there are three parties in the three-tier cloud marketplace: application users, SaaS providers and IaaS providers. Application users can get access to applications provided by SaaS providers. SaaS providers rent cloud resources from IaaS providers to run their applications and serve their customers. We assume that there are some SaaS providers who deliver workflow applications to their customers. That means these SaaS providers need to lease a collection of cloud resources to execute all tasks contained in workflows. In an open public cloud environment, the cloud market is diverse and full of uncertainty. Specifically, there are lots of different types of computing resources in the form of VM instances offered by IaaS providers with varying prices and configurations. Also the performance of VMs is always fluctuant because of the change of time and workloads. Furthermore, from the perspective of SaaS providers, their workflow applications normally have some constraints, such as deadline, task dependencies and hardware configuration. In order to maximize their profits in the long run, SaaS providers need to select the appropriate IaaS service for each task to execute their workflow applications profitably.

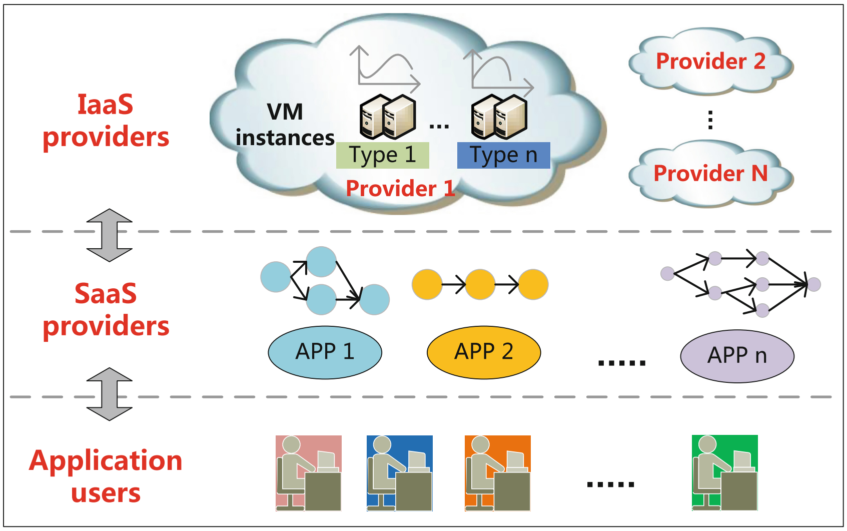

Fig. 1. Three-tire cloud marketplace

3.2 Problem Formulation

Workflow Model. In general, a workflow application is represented as a directed acyclic graph (DAG) $WA = (T, E)$. The finite set $T = \{T_1, \ldots, T_N\}$ denotes N tasks in the workflow. $E = \{E_{i,j} | 1 \leq i,j \leq N, i,j \in N^+\}$ is the set of directed edges which describes the precedence constraints between tasks. An edge $E_{i,j} = (T_i, T_j)$ states that T_j is the child task of T_i and it cannot be executed until all of its parent tasks have been completed. In a given workflow, the task without any parent tasks is called a start task T_{start} and the task without any successors is called an end task T_{end}. We assume that there is only one T_{start} and T_{end} in our model. *Data* is a $N \times N$ matrix where

$Data(T_i, T_j)$ is the amount of data that required to be transmitted from T_i to T_j. Each task is expected to be executed on one or more VM instances.

A Workflow application WA generally has some constraint $Constraint(WA) = \{GC, TC\}$. The former set $GC = \{deadline, budget\}$ defines the maximum makespan and cost of the whole workflow that users would like to accept. The later set $TC = \{TC_1, \ldots, TC_N\}$ contains N subsets which describe the restrictions of tasks. $TC_i = \{parallelism_i, \mathrm{CPU}_i, \mathrm{memory}_i\}$ denotes the maximum parallelism degree and the minimum configuration requirements of VM for task T_i.

Cloud Resources Model. The cloud market has M IaaS services $Resources = \{VM_1, \ldots, VM_M\}$. Each of them defines one type of VM instances $VM_i = \{DataCenter_i, eP_i, inP_i, outP_i, CPU_i, memory_i, bandwidth_i\}$, where $DataCenter_i$ identifies the location of VM_i, eP_i is the hourly execution cost, inP_i and $outP_i$ are the cost of transferring and receiving per unit of data from other VMs, and other attributes denote the VM capacity. Due to the task features and VM configurations, task runtime on different types of VMs is varying. Moreover, for one task, its runtime is not constant even though it is executed on the same VM. This is because the VM performance is also affected by its data center's workloads. There are K workload conditions $W = \{W_1, \ldots, W_K\}$ and the probability matrix for $DataCenter_i$ is $WP_i = \{P_i^1, \ldots, P_i^K\}$, where $\sum_{j=1}^{K} P_i^j = 1$. Supposing $VM(T_i)$ and $VMnum(T_i)$ represent the VM type and number that is scheduled to T_i, and $W(VM_j)$ is the current workload condition of $DataCenter_j$.

Task Execution Time and Cost. Let $\mathrm{basic}ET(T_i, VM_j, W_k)$ denote the execution time of task T_i on one VM_j instance when the data center condition is W_k. As some tasks can be executed concurrently on the same type of VMs, the actual execution time is:

$$ET(T_i, VM(T_i), W(VM(T_i))) = \mathrm{basic}ET(T_i, VM(T_i), W(VM(T_i)))/VMnum(T_i) \tag{1}$$

Generally, it is assumed that VM instances are charged hourly, thus the task execution cost can be calculated by:

$$\begin{aligned} &EC(T_i, VM(T_i), W(VM(T_i))) \\ &\quad = VM(T_i).eP * \lceil ET(T_i, VM(T_i), W(VM(T_i))) \rceil * VMnum(T_i) \end{aligned} \tag{2}$$

Data Transfer Time and Cost. Let $Bandwidth$ is a $M \times M$ matrix, in which $Bandwidth(VM_i, VM_j)$ is the bandwidth between VM_i and VM_j. For a pair of consecutive tasks, the data transfer time and cost between them are defined by (3) and (4) respectively.

$$\mathrm{TT}(T_i, T_j) = \frac{Data(T_i, T_j)}{Bandwidth(VM(T_i), VM(T_j))} \tag{3}$$

$$TC(VM(T_i), VM(T_j)) = Data(T_i, T_j) * (VM(T_i).outPrice + VM(T_j).inPrice) \tag{4}$$

Workflow Application Cost. The whole workflow cost is the total expense for all tasks execution, which contains the tasks' execution cost and data transfer cost. The computational formula for each task's cost is:

$$\text{Task\_cost}_i = EC(T_i, VM(T_i), W(VM(T_i))) + \sum_{j \in children(T_i)} TC\big(VM(T_i), VM\big(T_j\big)\big) \quad (5)$$

Thus the total cost of workflow application is $\text{WA}_{\text{cost}} = \sum_{i=1}^{N} \text{Task\_cost}_i$.

Workflow Makespan. Let $ST(T_i)$ and $ET(T_i)$ are the earliest start time and finish time of task T_i. For the start task T_{start}, we have:

$$ST(T_{start}) = 0 \quad (6)$$

$$ET(T_{start}) = ST(T_{start}) + ET(T_{start}, VM(T_{start}), W(VM(T_{start}))) \quad (7)$$

For the other tasks, their ST and ET values can be computed recursively:

$$ST(T_i) = \max_{T_j \in parents(T_i)} \big\{ET\big(T_j\big) + TT\big(T_j, T_i\big)\big\} \quad (8)$$

$$ET(T_i) = ST(T_i) + ET(T_i, VM(T_i), W(VM(T_i))) \quad (9)$$

Therefore, the total workflow execution time is $\text{WA}_{makespan} = ET(T_{end})$.

Cost-optimal Scheduling Problem. SaaS providers derive their profits from the margin between the revenue obtained from customers and the expense of renting cloud resources from IaaS providers. In order to maximize profits, the best policy is reducing the workflow application cost at the most and guaranteeing the makespan within customers' deadline constraints. Therefore, this problem can be formally defined as follows:

$$\begin{cases} min & WA_{cost} \\ s.t. & WA_{makespan} \leq deadline \\ & VMnum(T_i) \leq TC_i.parallelism_i \\ & VM(T_i).CPU \geq TC_i.CPU_i \\ & VM(T_i).memory \geq TC_i.memory_i \end{cases} \quad (10)$$

4 Workflow Scheduling Approach Using Q-Learning

4.1 Theoretical Foundations

Reinforcement learning (RL) [10] is a widely used method in the field of machine learning. Faced with an unknown environment, an agent without prior knowledge can improve its behavior based on the interactions with its environment. In general, reinforcement learning problems can be modeled as Markov Decision Processes (MDPs)

$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. $\mathcal{S}$ is a set of environment states and $\mathcal{A}$ is a set of actions that an agent can take. $\mathcal{P}$ defines the probability of the state transition, where $\mathcal{P}_a(s, s') = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$ represents the possibility of action a in state s at time t will move to a new state s' at time $t + 1$. $\mathcal{R}$ is a reward function which yields a reward for each state transition. $\gamma \in [0, 1]$ is a discount factor which is used to balance the influence of present reward and future rewards. The goal of RL agents is to find the optimal behavior for each state to maximize the accumulated discounted rewards in the long-term.

Q-learning [11] as a model-free reinforcement learning algorithm which can be used to find optimal policies by learning from previous decision-making experiences. It uses Q function to approximate the long-term benefit of each state-action pair <s, a> by updating Q values during learning processes. The update rule is defined in (11), where $\alpha \in [0, 1]$ is the learning rate and $\gamma \in [0, 1]$ is the discount factor. It will be updated based on the next state s' and obtained reward $R(s, a, s')$.

$$\mathrm{Q}(s, \mathrm{a}) = \mathrm{Q}(\mathrm{s}, a) + \alpha[R(s, a, s') + \gamma max_{a'} Q(s', a') - \mathrm{Q}(\mathrm{s}, \mathrm{a})] \tag{11}$$

4.2 Level-Based Workflow Scheduling Approach

In general, the workflow scheduling problem is a multidimensional, multi-choice, multi-constraint optimization problem (MMMO), which is NP-hard to solve. On this basis, in the dynamic cloud market, performance of IaaS services is always filled with uncertainty. For SaaS providers, it is hard to predict the task execution time on different VMs at a certain time in advance. Therefore, traditional heuristic approaches such as genetic algorithms do not perform well. By contrast, we can model the IaaS service composition problem as a MDP and solved it by Q-learning algorithm. Our level based scheduling approach involves two phases: initialization phase and decision-making phase. We will describe them respectively.

4.2.1 Initialization Phase

In this phase, workflow application tasks will be classified into different execution levels. Execution level *EL* represents the task scheduling sequence in this workflow. All tasks belonging to the same level will be scheduled at the same time during the decision-making phase. As we mentioned before, for any task T_i, only when all its parent tasks have been finished, it can be executed. According to this principle, the execution level allocation algorithm is shown below:

Algorithm 1: Execution level allocation algorithm

```
1.  $T_{start}.EL \leftarrow 1$
2.  add $T_{start}$ to $FinishTasks$
3.  $T_{current} \leftarrow T_{start}$
4.  while $T_{current} != T_{end}$
5.      for $\forall T_i \in children(T_{current})$
6.          if $T_j \in FinishTasks,$ where $\forall T_j \in parent(T_i)$ then
7.              $T_i.EL \leftarrow T_{current}.EL + 1$
8.              add $T_i$ to $ExecuteTasks$
9.          end if
10.     end for
11.     add $ExecuteTasks$ to $FinishTasks$
12.     $T_{current} \leftarrow FinishTasks(T_{current}).next$
13. end while
```

4.2.2 Decision-Making Phase

During this phase, task scheduling plans will be decided and executed according to the sequence of *EL*. When the whole workflow application has been scheduled, it means one decision-making process has been completed. Such a learning process will be repeated many times until the near-optimal scheduling plan has been found. In this part, we will first describe the key elements of this Q-learning problem and then present the algorithm in decision-making phase.

A. State Space and Actions

The state space S contains all possible states that SaaS providers can experience. We define each state as a 3-tuple: $s = (EL, datacenter, time)$, where:

- *EL* is the current execution level that will be scheduled;
- *datacenter* is a set which records the VM locations for all tasks in the last *EL* level;
- *time* is the total execution time of the uncompleted workflow until now.

Given a state $s \in S$, the corresponding action set $A(s)$ includes all available actions that can be taken. Supposing there are *C* tasks in current *EL* level, each action at state s is expressed as a collection shown in (12), where $VMid_c$ and $VMnum_c$ are the VM type and number allocated to task T_c.

$$a = \{(VMid_1, VMnum_1), \ldots, (VMid_C, VMnum_C)\} \tag{12}$$

B. Reward Function

Because the goal of application scheduling algorithm is to find the cost-optimal choices for the whole workflow, the reward function. $R(s, a, s')$ at state s is given by the following equations:

$$\begin{cases} R_{success} - \sum\limits_{T_i.EL=S.EL} \text{Task\_cost}_i, \mathit{if}\ s.EL = EL_{final}\ \mathit{and}\ s'.time \leq deadline \\ R_{fail} - \sum\limits_{T_i.EL=S.EL} \text{Task\_cost}_i, \mathit{if}\ s.EL = EL_{final}\ \mathit{and}\ s'.time > deadline \\ - \sum\limits_{T_i.EL=s.EL} \text{Task\_cost}_i, \mathit{else} \end{cases} \tag{13}$$

C. Action Selection and State Transition

At a decision time T_i, we adopt the ε-Greedy policy to choose an action for current state s. It means that the optimal action based on current knowledge will be chosen with a high probability (1 − ε), while a random action instead of the best one be taken with a low probability ε, where $\varepsilon \in [0, 1]$. Once SaaS performs a specific action a, its state will change from s to s'. The corresponding transitions can be described as follows:

$$\begin{cases} s'.EL = s.EL + 1 \\ s'.datacenter = \{a.VMid_c.DataCenter | c = 1, \ldots, C\} \\ s'.time = s.time + \max\limits_{T_i.EL=s.EL} ET(T_i) \end{cases} \tag{14}$$

where $s'.EL$, $s'.datacenter$ and $s'.time$ are elements of new state s'. Meanwhile, the Q value of state-action pair <s, a> will be updated based on (11).

D. Q-learning based IaaS service composition algorithm

Algorithm 2: Q-learning based IaaS service composition algorithm

```
1.  Loop (for each scheduling decision-making process)
2.    Choose action $a$ from $A(s)$ using ε-Greedy policy
3.    Rent appointed VMs to execute tasks according to
      action a
4.    Move to the new state $s'$ according to (14)
5.    if $s'$ is a new state then
6.       add $s'$ into state set $S$
7.       For $\forall a' \in A(s')$, initialize Q-values $Q(s', a')$
8.    end if
9.    calculate reward $r \leftarrow R(s, a, s')$ using (13)
10.   update Q(s,a) using (11)
11.   s $\leftarrow s'$
12. end Loop
```

5 Experiment Results and Analysis

In this section, we evaluate our level based workflow scheduling approach through a series of experiments. We first describe the basic experiment settings and then present the relevant results and analysis.

5.1 Experimental Setup

Attributes of Cloud Resources. We simulate a dynamic cloud environment which involves two IaaS providers. Each of them has one data center and offers three kinds of VM types with different configurations and prices. Data transfer speeds and costs between different data centers are varying. In terms of the performance of data centers, we define two kinds of workload conditions: normal condition (NC) and busy condition (BC). For a task, its execution time under the busy condition is 1.5 times of that under the normal condition on the same VM instance. The probability of each condition in different data centers is different. Let $p = (p_1, p_2)$ denote the probability of BC in two data centers. We test 5 cases $\{(0, 0), (0.2, 0.4), (0.3, 0.7), (0.7, 0.9), (1, 1)\}$ in our experiments, which implies the data centers' conditions are both normal, both a little busy, one a little busy and one busy, both busy, both very busy respectively.

Workflow Application Settings. In our experiment, we test 8 cases with different number of tasks in the workflow. The number of tasks is varied from 3 to 10. Each task excepting T_{end} has at least one child task. The average number of child tasks for tasks is set to 2. Task execution time on different VMs and the amount of transmitted data are randomly generated in [10 min, 100 min] and [5 GB, 20 GB] respectively. The deadline and budget constraints are set dynamically based on the value bounds of workflow makespan and monetary cost.

Q-Learning Parameters. We set the the values of learning rate α and discount factor γ to 0.6 and 0.8 respectively. The initial value of parameter ε is 0.2. We decrease its value gradually so as to reduce explorations and obtain the best renting decisions with a high probability. As for the reward function, we set $R_{success} = 1$ and $R_{fail} = -5$.

Baseline Algorithms. We compare our algorithm (QA) with alternative five service selection strategies: optimal strategy (OS), random strategy (RS), conservative strategy (CS), adventurous strategy (AS) and guess strategy (GS). OS strategy is an ideal baseline, which assumes that all information about the cloud market (e.g. task execution time and data center conditions at a given time) is known in advance. RS is a simple strategy in which a random workflow scheduling plan is chosen. Both CS and AS give constant workflow scheduling plans each time. The difference is that they are based on different data center conditions. Specifically, CS gives the plan that is the optimal one under data centers' busy condition (i.e. $p = (1,1)$), while AS gives the plan that is the optimal one under data centers' normal condition (i.e. $p = (0,0)$). GS strategy assumes that SaaS providers guess the market condition is $p = (0.3, 0.7)$.

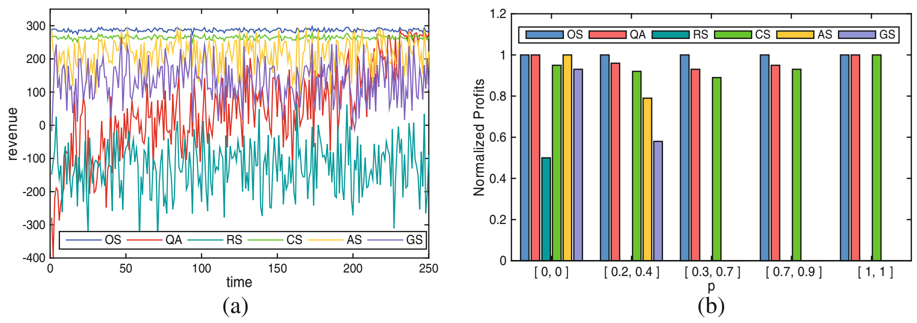

Fig. 2. (a) Performance comparison of six strategies. The profit of each time slot equals the sum profits of 20 consecutive service composition decision-making rounds. (b) Profits comparison under different data center conditions. If the real normalized value is less than 0, we set it to 0 in this figure.

5.2 Experiments and Results

In our experiments, there are 5 cases with respect to data center conditions and 8 cases regarding the number of tasks. We test our workflow scheduling approach and other 5 baselines under these scenarios. For each scenario, we do experiments 10 times with different stochastic workflows. Because reducing workflow application cost to maximize profits is the ultimate goal of SaaS providers, we focus on the service composition profits obtained by applying different scheduling strategies in each round. Figure 2a shows the comparison results of 6 strategies for one experiment. In this scenario, the number of workflow tasks is 6 and the data center's condition is p = (0.2, 0.4). It is clear that the workflow scheduling plans generated by QA gain low profits at the early stage of decision-making rounds. However, the profit curve grows gradually and then reaches a high and relatively stable level which is close to the maximum profit curve produced by OS strategy. By contrast, workflow scheduling plans generated by RS strategy and GS strategy always get low profits during the whole period. Furthermore, we also compare QA with other 5 strategies when its final near-optional scheduling plans have been obtained. We calculate the total profits in last 100 rounds of our QA and other baselines respectively. And then we normalize results based on the profit obtained by OS. Figure 2b gives the profit comparisons under different conditions of data centers. It can be seen that no matter what the condition is, QA is able to generate high profits which are close to the maximum profits. In terms of the experiments in other scenarios, we can get similar results as shown in Fig. 2. Due to the space limitation, they are not included in this paper.

6 Conclusion

In our work, we proposed a workflow application scheduling approach based on the Q-learning algorithm to help SaaS providers make near-optimal service selection decisions in a dynamic and stochastic cloud environment. Through a series of experiments, we demonstrated the effectiveness of our approach and compared it with other baseline strategies in different market settings.

Acknowledgments. The authors would like to acknowledge the support provided by the National Natural Science Foundation of China (61402263, 91546203), the National Key Research and Development Program of China (2016YFB0201405), the Key Research and Development Program of Shandong Province of China (2017CXGC0605), the Shandong Provincial Science and Technology Development Program (2016GGX106001, 2016GGX101008, 2016ZDJS01A09), the Natural Science Foundation of Shandong Province (ZR2014FQ031), the Fundamental Research Funds of Shandong University (2016JC011), the special funds of Taishan scholar construction project, and China Scholarship Council (201606220190).

References

1. Wu, D., Rosen, D.W., Wang, L., Schaefer, D.: Cloud-based design and manufacturing: a new paradigm in digital manufacturing and design innovation. Comput. Aided Des. **59**, 1–14 (2015)
2. Smanchat, S., Viriyapant, K.: Taxonomies of workflow scheduling problem and techniques in the cloud. Future Gener. Comput. Syst. **52**, 1–12 (2015)
3. Masdari, M., ValiKardan, S., Shahi, Z., Azar, S.I.: Towards workflow scheduling in cloud computing: a comprehensive analysis. J. Netw. Comput. Appl. **66**, 64–82 (2016)
4. Fakhfakh, F., Kacem, H.H., Kacem, A.H.: A provisioning approach of cloud resources for dynamic workflows. In: 8th IEEE International Conference on Cloud Computing, pp. 469–476. IEEE (2015)
5. Wei, Y., Pan, L., Yuan, D., Liu, S., Wu, L., Meng, X.: A cost-optimal service selection approach for collaborative workflow execution in clouds. In: 20th IEEE International Conference on Computer Supported Cooperative Work in Design, pp. 351–356 (2016)
6. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17358-5_7
7. Jungmann, A., Kleinjohann, B.: Learning recommendation system for automated service composition. In: 2013 IEEE International Conference on Services Computing, pp. 97–104 (2013)
8. Yan, Y., Zhang, B., Guo, J.: An adaptive decision making approach based on reinforcement learning for self-managed cloud applications. In: 2016 IEEE International Conference on Web Services, pp. 720–723 (2016)

9. Barrett, E., Howley, E., Duggan, J.: Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concur. Comput. Pract. Exp. **25**(12), 1656–1674 (2013)
10. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
11. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)