Journal of Ambient Intelligence and Humanized Computing (2021) 12:10823–10835
https://doi.org/10.1007/s12652-020-02884-1

ORIGINAL RESEARCH

Workflow scheduling based on deep reinforcement learning in the cloud environment

Tingting Dong[1] · **Fei Xue**[2] · **Chuangbai Xiao**[1] · **Jiangjiang Zhang**[1]

Received: 24 September 2020 / Accepted: 23 December 2020 / Published online: 9 January 2021

Abstract
As a convenient and economic computing model, cloud computing promotes the development of intelligence. Solving the workflow scheduling is a significant topic to promote the development of the cloud computing. In this work, an Actor-Critic architecture is utilized to solve this problem achieving the task executive time minimization under the task precedence constraint. It is similar to the list-based heuristic algorithm which includes the task prioritizing phase and task allocation phase. However, the results of the two phases interact with each other. In the task prioritizing phase, given a workflow represented as the data communication time matrix and task computation time matrix, a distribution over different task permutations by the improved Pointer network can be predicted. Then, the heuristic algorithm based on the HEFT achieves the task allocation to get the task executive time. Using negative task executive time as the reward signals, the model parameters by a policy gradient method in the first phase can be optimized. The simulation experiment is done from the task executive time, and the results shows that the workflow scheduling by the deep reinforcement learning is more effective comparing with other four single objective heuristic algorithms.

Keywords Cloud computing · Workflow scheduling · Deep reinforcement learning · Actor-Critic

1 Introduction

In the process of transforming the industrial economy into the digital economy, cloud computing as a distributed computing model plays a vital role (Yuan et al. 2010). Gartner, a authoritative research institution, released the global cloud computing market research data, which showed that the global public cloud service market will reach $331.2 billion in 2022 and the size and growth of the cloud services industry will be nearly three times that of the overall IT service (Market Share: IT Services, Worldwide 2019).

Cloud computing is a promising utility-oriented computing paradigm, and it provides users computing resources over the Internet to execute applications submitting by users (Bodrow 2017; Sun et al. 2020; Cai et al. 2020c). The cloud computing platform consists of a cluster of distribution computing resources to execute large-scale and complex applications. Workflow can also be used to describe the data-intensive application. A workflow expressed as the directed acyclic graph (DAG) is a parallel and sequence model, consisting of a set of tasks with their precedence relationship, that is, a task can't be executed until all of its immediate predecessors are completed. Some large-scale scientific and engineering computing problems in astronomy, physics, bioinformatics and many other fields (Juve et al. 2013) can be abstracted as workflow models. These scientific applications usually have the characteristics of data and computing intensive. A high-performance cloud environment is offered to the large-scale computing requirements of workflow. The completion of a workflow usually requires the cooperation of multiple heterogeneous servers. Workflow scheduling (Karpagam et al. 2020) can map all tasks in a workflow into the available cloud computing resources to achieve certain goals under some constraints, which is a challenging problem in the cloud computing environment.

It is not a feasible approach to schedule each task manually due to the complex and large-scale workflow

✉ Fei Xue
Xuefei2004@126.com

Tingting Dong
dongtingting2019@163.com

[1] Faculty of Information Technology, Beijing University of Technology, Beijing, China

[2] School of Information, Beijing Wuzi University, Beijing, China

construction. Many research scholars have devoted themselves to the field of workflow scheduling with the important performance metrics of solution quality and computing complexity which are also two conflicting goals. Heuristic algorithms (Faragardi et al. 2020; Rajasekar and Palanichamy 2020; Liu 2020) are a popular way, which can be easy to implement and have low computational complexity to get a local optimal solution. However, the heuristic algorithms lack robustness that work only on specific issues. What's more, the solution quality deteriorates with the increasing number of problem instances. Another popular way to solve the problem is meta-heuristic algorithms (Cui et al. 2020; Cai et al. 2020a; Nedjah et al. 2020) whose main disadvantage is that they need high computational cost.

In this work, the purpose is to design the method to tackle the workflow scheduling from the perspective of solution quality and computational complexity. The emergence of Machine learning (ML) (Hopfield and Tank 1985; Cai et al. 2020b) promotes the development of artificial intelligence (AI) (Hassan et al. 2020a, b) and can solve more complex and high-dimensional issues than the tradition methods. As the major role of ML, Reinforcement learning (RL) (Sutton and Barto 1998) learns the optimal policy through data generated from interaction with the environment, which is necessary for scheduling problem without high quality labels. Deep reinforcement learning (DRL) (Mnih et al. 2015) combines nonlinear learning in deep learning (Cui et al. 2018) and decision learning in reinforcement learning, which can be applied in many fields. In this work, we utilize the Actor-Critic (Barto et al. 1970) which is a kind of DRL to tackle the workflow scheduling problem in cloud environment. We make an improvement on the classical list-based heuristic which includes task prioritizing phase and task allocation phase. In the proposed DRL architecture, the two phases are combined and interact with each other. The training model is obtained by training the off-line data set. The trained model can be directly used to solve practical problems, which can reduce the calculation time. The three major contributions of this work are:

1. An Actor-Critic architecture is designed to tackle the workflow scheduling problem aiming at minimizing the task execution time under the task precedence constraint.
2. The improved Pointer network as the Actor network, called the P-Network model, is used to predict the task sort distributed which can be as the input of task allocation phase, and a simple heuristic algorithm is designed to select the server from the the perspective of the task precedence relationship and computational complexity, which is reward to adjust the task sort.
3. Comparing with other four single objective heuristic algorithms, the performance of the proposed algorithm in this work is effective in aspect of the average task executive time (makespan) and the efficiency.

The rest of this work is organized as below. Some related works about the list-based heuristic algorithms and deep reinforcement learning are discussed and analyzed in Sect. 2. Problem description and formulation are expressed in Sect. 3. Deep reinforcement learning architecture for the workflow scheduling is proposed in Sect. 4. In Sect. 5, we make simulation experiments to evaluate the performance of our algorithm. Finally, we give conclusions and future works in Sect. 6.

2 Related work

Scheduling is a fundamental problem in many fields (Zhang et al. 2020; Kumar and Giri 2020). DAG-based workflow scheduling problem is proved to be NP-hard, and it is hard to find the optimal solution. In this work, the related work of the workflow scheduling is given for the list-based heuristic algorithms and reinforcement learning.

List-based heuristic algorithm is a popular heuristic algorithm to solve the workflow scheduling problem, consisting of task prioritizing phase and task allocation phase. One of the well-known list-based heuristic algorithm is the heterogeneous earliest-finish-time (HEFT) (Topcuoglu et al. 2002) which has low computational complexity. Task prioritizing phase is that all tasks in the workflow form a sequence according to certain rules which can determine the task allocation order in the second phase. The upward rank was a common method to get the task sequence of a workflow, which utilized the average execution time and communication time of the task to calculate the sorting value traversing each task from bottom to up. Algorithms using this method include HEFT, Lookahead (Bittencourt et al. 2010) and predict earliest finish time algorithm (PEFT) (Arabnejad and Barbosa 2014), etc. Critical-path-on-a-processor (CPOP) (Topcuoglu et al. 2002) utilized the sum of the average execution time and communication time of tasks to determine the task priority, and defined the critical path using the longest path from the starting task to the finishing task. Tasks on the critical path and other tasks adopt different allocation strategies. Task allocation phase is that tasks are assigned to servers in turn to satisfy certain goals according to the obtaining task sequence in the first phase. The algorithm of HEFT selected the server with the earliest-finish-time (EFT) to allocate the task. Considering the effect of succeed task execution time on the current scheduling decision, Bittencourt et al. (2010) proposed the Lookahead algorithm based on HEFT to improve the optimization, but the time complexity of the algorithm was higher than that of HEFT. For the high complexity of Lookahead, Arabnejad

and Barbosa (2014) considering the effect of the current task execution time on the next decision proposed PEFT, but the algorithm had the poor optimization when the quantity of parallel tasks is large. The task prioritizing phase is important for the second phase. However, the first phase is independent of the second phase in the list-based heuristic algorithms. Both phases are vital for the workflow scheduling and should interact with each other. In this work, the two phases adjust each other to achieve the optimization by deep reinforcement learning.

Since the concept of ML was proposed, it has aroused intense research in many fields (Khan et al. 2020; Cui et al. 2020). DRL is already successfully used to solve combinatorial optimization problems, bin packing problems, job scheduling problems, etc., which shows its huge potential and inspires us to solve the workflow scheduling problem. Dong et al. (2020) proposed a deep Q learning workflow scheduling architecture to minimize the makespan. Aiming at the makespan and load balance, Tong et al. (2019) utilized the deep Q-learning algorithm to solve the multi-objective workflow scheduling problem. Combining *Q_learning* and HEFT, Tong et al. (2020) utilized *Q_learning* to achieve the task sort and EFT to finish the task allocation aiming at minimizing the makespan. However, the reward in *Q_learning* was required by the upward rank value, which caused the separation of the two phases. Similarly, Asghari et al. (2020) combined *Q_learning* and the heuristic algorithm to solve the multi-objective online workflow scheduling. However, *Q_learning* is limited in solving the large-scale workflow scheduling problem. In this work, we utilized the DRL instead of the *Q_learning* algorithm to achieve the task sort.

In this work, we propose a workflow scheduling strategy by DRL. Given any the data communication time matrix and the task computation time matrix and the certain characteristics of tasks, the distribution over different task sorts can be found utilizing the constructed neural network. And the task sort can be as the input of task allocation phase. In turn, the result of server selection using the proposed simple heuristic algorithm can be the reward to adjust the task sort.

3 Problem description and formulation

3.1 The description of the cloud workflow scheduling system

Cloud computing is a new type of computing model oriented to numerical and information processing, which integrates distributed computing, Internet technology, large-scale resource management and other technologies, and makes reasonable use of computing power and resources. In essence, it can be regarded as a large-scale heterogeneous parallel distributed computing system. As the popular cloud computing service model, IaaS(Infrastructure as a Service) can provide clients with many heterogeneous service resources to accomplish their various demands.A cloud platform based on IaaS is utilized in this work. Figure 1 shows the cloud workflow scheduling framework in this work.

The cloud workflow scheduling system is made up of the IaaS-based platform and clients. For clients, the computing resources in the cloud are infinite. They present their application requirements (workflow), which is composed of a set of task units, to the cloud platform and can obtain the computing resources. The IaaS-based platform consists of numerous heterogeneous servers, which contain the different computational capabilities. The function of cloud platform is to fulfill the clients demands.

Used symbols and notations referred to the workflow scheduling model are in Table 1.

3.2 Problem formulation

Some quantities used in this work need to be defined and explained. A DAG graph can express a workflow, which is comprised of task nodes T, where $T = t_1, t_2, \ldots, t_m$ represents m tasks in a workflow, and edges D, where $D = \{d_{ij} | i \neq j, and i, j \in \{1, \ldots, m\}\}$ indicates the conveying data value and the precedence constraint relationship between t_i and t_j. TP_{ij} is utilized to express the precedence

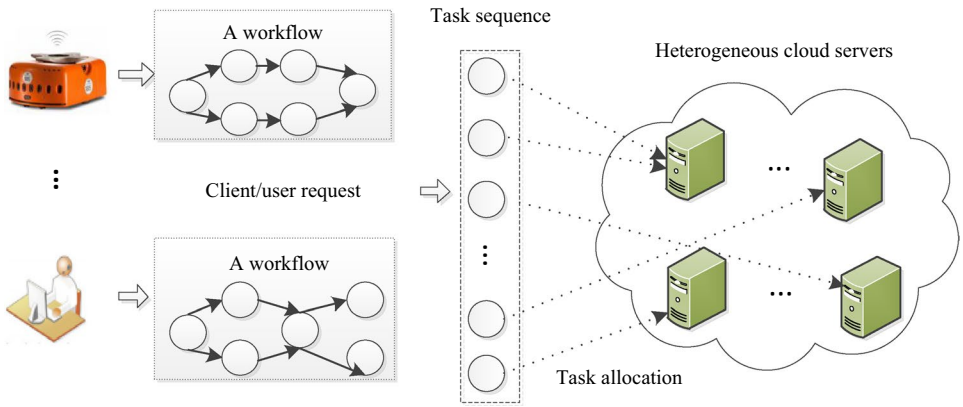


Fig. 1 The cloud workflow scheduling framework

Table 1 Symbols and notations referred to the workflow scheduling model

Notation	Definition
n	Server numbers
m	Task numbers
T	The set of tasks, $T = t_1, t_2, \ldots, t_m$
V	The set of tasks, $V = v_1, v_2, \ldots, v_n$
$TCT(t_i, v_j)$	The task computation time of t_i on v_j
$DCT(t_i, t_j)$	The task communication time between t_i and t_j
TP_{ij}	The precedence relationship between t_i and t_j
α_{ij}	The data metastasis rates between v_i and v_k
d_{ij}	The conveying data value from t_i to t_j
$SET(t_i, v_j)$	The start execution time of t_i on v_j
$FET(t_i, v_j)$	The finish execution time of t_i on v_j

relationship between any two tasks. $TP_{ij} = 1$ represents that the immediate predecessor of the t_i is the t_j. Otherwise, TP_{ij} is 0. The task t_{entry} is the starting task without any predecessor node or father node, and the task t_{exit} is the lasting task without any successor node or son node. Data communication time (DCT) represents conveying the data between two tasks with the dependency relationship, the value of which is relevant to the conveying data value and metastasis rates and shows on each edge in Fig. 2. Task computation time (TCT) represents the time for different servers to execute a task, shows in Fig. 3.

Workflow is the combination of parallel and sequence construction. Each task in a workflow can be only allocated to one server, and a server can only execute one task at a time. DCT is generated only when two tasks with the dependency relationship are assigned on different servers. Otherwise, DCT between two tasks is zero. Setting the server

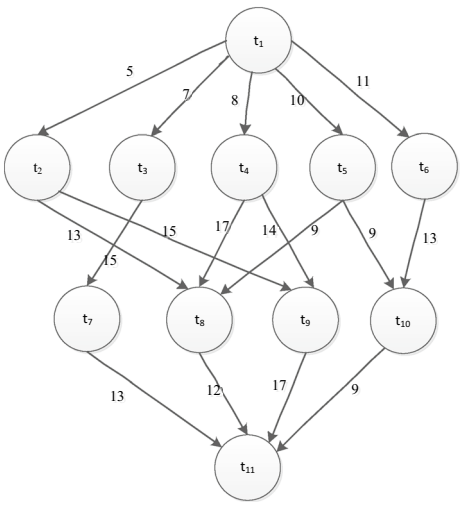


Fig. 2 Example of a DAG-based workflow

Task	Server1	Server2	Server3
t_1	9	13	18
t_2	22	19	41
t_3	15	27	39
t_4	32	23	43
t_5	34	42	46
t_6	19	38	44
t_7	43	26	36
t_8	41	38	33
t_9	38	29	11
t_{10}	56	27	62
t_{11}	18	28	33

Fig. 3 Time computation time matrix

set $V, V = v_1, v_2, \ldots, v_n$, and data metastasis rates between v_l and v_k as α_{lk} , we can define the Data communication time between t_i and t_j by

$$DCT(t_i, t_j) = \begin{cases} \frac{d_{ij}}{\alpha_{lk}}, & TP_{ij} = 1 \cup t_i \ in \ v_l \ and \ t_j \ in \ v_k (l \neq k) \\ 0, & otherwise \end{cases} \tag{1}$$

3.3 Model building and the definition of elements in deep reinforcement learning

In this work, we focus on the executing time minimization of a workflow, that is the makespan minimization, to design a reasonable workflow scheduling algorithm based on deep reinforcement learning. By analyzing the factors that affect the execution time of tasks, the elements in deep reinforcement learning are defined.

Figure 4 shows the process of task allocation. For a workflow, the allocation of the starting task t_{entry} is the basic of the whole scheduling, the start execution time of which is zero. The allocation of parallel tasks (tasks without dependencies) is the key to the optimization of workflow scheduling. With the increasing of parallelism degree, the execution time of the whole workflow is shorten. Therefore, it is of great significance to improve the parallelism of task execution to optimize the workflow scheduling. The description of the task's start and finish time on the server provides the basis for other tasks' allocation in the whole workflow. For the task t_i, the finish time of immediate predecessors and the DCT between immediate predecessors and t_i and the task earliest execution time on the server jointly determine the start execution time of the task t_i. The start execution time of task t_i on the server v_j can be defined as

$$SET(t_i, v_j) = max(\max_{t_p \in Pre(t_i), h \in [1,n]} (FET(t_p, v_h) + DCT(t_p, v_h)), \max_{t_l \in A(v_j)} FET(t_l, v_j)) \tag{2}$$

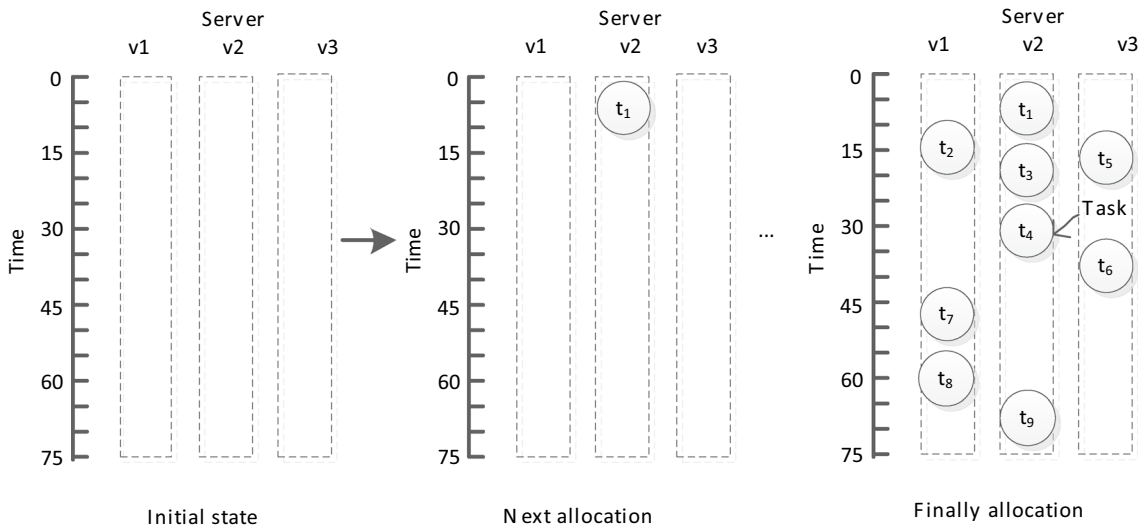


Fig. 4 The process of task allocation

Where $FET(t_p, v_h)$ represents the finish execution time of task t_p on the server v_h. $Pre(t_i)$ is the immediate predecessor set of t_i. $A(v_j)$ represents the task sets allocated on the server v_j.

In this work, once the task is assigned to the server, it will not be interrupted until the task is finished. Therefore, the start execution time and TCT on the server determine the finish time of the task t_i. The finish time of task t_i on the server v_j can be defined as

$$FET(t_i, v_j) = SET(t_i, v_j) + TCT(t_i, v_j) \tag{3}$$

Because multiple servers work in parallel, the execution time of the whole workflow depends on the finish time of the task on the server with the longest working time. In this work, the objection function of the workflow scheduling problem is to minimize the workflow makespan. Then the objection function can be defined by.

$$F = \max_{j\in[1,n]} \left(\max_{t_l \in A(v_j)} FET(t_l, v_j) \right) \tag{4}$$

$$minmakespan = minF \tag{5}$$

A deep reinforcement learning architecture is utilized to solve the workflow scheduling problem. The state space and action space in the deep reinforcement learning are defined by the established mathematical model and the known quantities.

State space definition The characteristics of tasks can be included in the state space, consisting of DCT, TCT, the start and finish execution time of tasks in servers. The definition of state space is as follows.

$$\begin{aligned} S = \Big\{ & DCT(t_i, v_j), TCT(t_i, v_j), \\ & \forall_{t_i \in A(v_j)} \left[SET(t_i, v_j), FET(t_i, v_j) \right] \Big\}, \\ & i \in (1, \ldots, m), j \in (1, \ldots, n) \end{aligned} \tag{6}$$

Action space definition Utilizing the deep reinforcement learning to achieve task sorting, the final result is to get a task sequence. Therefore, tasks are regarded as actions, and a task is selected at each decision step. In addition, in order to improve the success rate of action selection and speed up the convergence speed of the algorithm, a *Mask* scheme is added to restrain the action selection at each episode. See Sect. 4.1 for details.

4 Deep reinforcement learning architecture for workflow scheduling

Value-based and policy-based iteration are two representative methods of reinforcement learning. The value-based iteration method Watkins and Dayan (1992) selects actions to obtain the optimal policy by estimating values of actions, including Q-Learning, Sarsa, temporal difference learning, etc. The policy-based iteration method obtains the optimal policy by adjusting the parameters in policies. Actor-Critic combines the policy-based iteration and value-based

iteration methods, which is an excellent deep reinforcement learning. Actor-Critic includes the Actor network and Critic network. Actor network is used to predict the policy probability distribution with parameters about different task sorts, whose network structure is the P-Network in this work. Critic network is used to estimate the expected value with parameters for a given state to evaluate the quality of the action policy. The Actor-Critic framework is used to get the optimal action policy by running multiple training episodes in this work.

The motivation to use the Actor-Critic architecture to solve workflow scheduling problems in cloud environment is as follows.

1. Inspiring by using DRL to solve the combinatorial optimization problem (Irwan et al. 2016), we use DRL to mix task prioritizing phase and task allocation phase, which can get better near-optimal solution than that two parts are carried out separately.
2. Using this proposed architecture to obtain the training model, we can get a near-optimal solution in a short time for a new workflow instant.
3. Unlike value-based iteration method, which limits the scale of the problem, utilizing this framework can solve high-dimensional and more complex issues.

4.1 Architecture of the neural network for the task sorting

Task prioritizing phase is to get the task allocation order as the input of task allocation phase. Pointer network (Oriol et al. 2015) is a kind of neural networks and is used to solve problems in many fields, including Traveling Salesman Problem (TSP), Convex Hull, Delaunay Triangulation, etc. It can achieve the input sequence sort with variable output dictionary size. Utilizing the principle of pointer network as the Actor network is proposed to obtain the task sort in this work.

Pointer network consists of two recurrent neural network (RNN): the encoder and the decoder. *DCT* and *TCT* are as the neural network input points X. But the set of task characteristics are as inputs which do not have fixed order in our model, we use the proposed neural network model in Mohammadreza et al. (2018) to directly use the embedded task characteristics instead of the encoder, which can simplify the computational complexity. At each decoding time step $t(t = 1, \ldots, n)$, the decoder uses attention mechanism Dzmitry et al. (2015) to point to an input as the output which is also the input of the next decoding time step $\pi(t)$. When all tasks are as the outputs, this process can be terminated. We aim to find a stochastic policy π with a different task sequence which can minimize the loss function by learning the parameters. The probability can be defined by the chain rule.

$$p(\pi|X) = \prod_{t=1}^{m} p(\pi_{t+1}|\pi_1, \pi_2, \ldots, \pi_t, X) \tag{7}$$

Attention mechanism is a neural network to provide the distribution over different outputs at every decoding time step t, which is helpful for pointer networks to force on the important input information and reduce the influence of unimportant information. First, the relevant information is extracted between the inputs x_i and the decoder hidden state h_t at decoding time step t, which can make better use of the input information and guide the $(t+1)-th$ time step to make a decision. A vector a_t can be used to express the relevant weight between every input x_i to the decoder hidden state h_t.

$$u_t^i = v_a^T tanh(W_1 x_i + W_2 h_t) \tag{8}$$

$$a_t = softmax(u_t) \tag{9}$$

$$p(\pi_{t+1}|\pi_1, \pi_2, \ldots, \pi_t, X) = a_t \tag{10}$$

Where using softmax function normalizes the u_t, and W_1, W_2 and v_a represent the trainable parameters.The proposed neural network can be called P-Network and shown in Fig. 5.

A *Mask* scheme is defined as the 0-1 list, which is used to generate feasible solutions and improve the success rate of the training. The scheme restrains the action selection from three aspects. First, considering that each task can only be selected once during a complete task sorting process, the value of the selected task position in the function is set to be 0. Then, considering that the task t_{entry} only has son nodes and the task t_{exit} only has father nodes, the t_{entry} is set as the first one and the t_{exit} is set as the last one in the task sequence. Finally, considering the dependency between tasks, all its father nodes have been selected when a task is selected. The pseudo code of the process of the *Mask scheme* is given in Algorithm1.

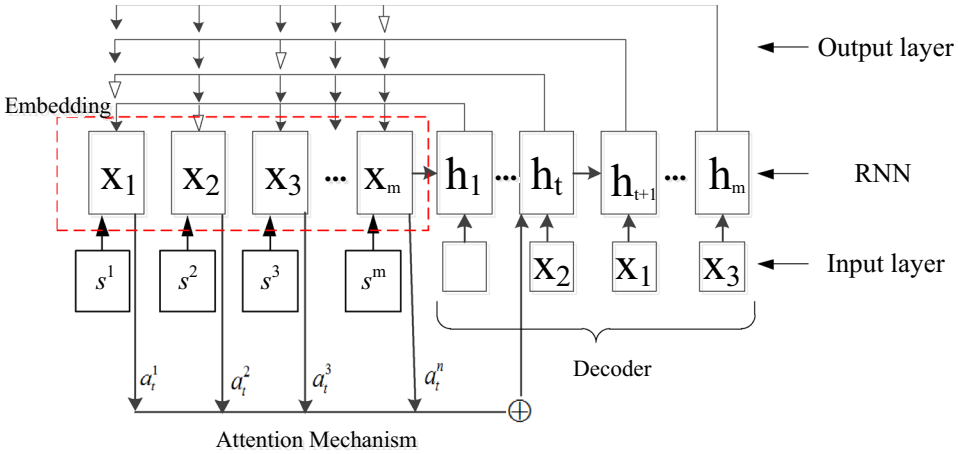


Fig. 5 Task sorting based on the P-Network model

Algorithm 1 The process of the Mask scheme

Input: the $1 * m$ list of $Mask$, all value in the list is 1.
Output: the processed list of $Mask$.

1: $mask = \text{ones}(m)$
2: **for** $i \leftarrow 1$ to m **do**
3: $\quad mask_1 = mask$
4: $\quad$ **for** $j \leftarrow 1$ to m **do**
5: $\quad\quad$ **if** $mask[j] = 1$ **then**
6: $\quad\quad\quad$ **for** $l \leftarrow 1$ to m **do**
7: $\quad\quad\quad\quad$ **if** $DCT(l, j) \neq 0$ and $mask[l] = 1$ **then**
8: $\quad\quad\quad\quad\quad mask_1 = 0$
9: $\quad\quad\quad\quad$ **end if**
10: $\quad\quad\quad$ **end for**
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad p = softmax(p + mask_1.log)$
14: $\quad chosen_{idx} \leftarrow \xi - greed$
15: $\quad \text{mask}[chosen_{idx}] = 0$
16: **end for**

In the Algorithm 1, the *Mask* list is initialized, and all value in the list is 1 (line 1). For each iteration, tasks, whose father nodes still are not allocated, are found among all tasks, and the value of those tasks is set to 0 (line 2–12). Then, the policy probability of unfeasible solutions is set to $-\infty$ using the log-probabilities (line 13). The next task is selected utilizing the ξ-*greed* policy (line 14). The Mask value of the selected task is set to 0 (line 15). Repeat the loop until all tasks are selected.

4.2 A heuristic algorithm based on the HEFT for the task allocation

Task prioritizing phase can get the distribution with different task sorts by the proposed neural network, then the task sort policy π can get. For the task allocation phase, tasks in the workflow are allocated to servers and the makespan of the whole workflow can be calculated. And the makespan can be as reward to make a feedback on the result of task sequencing. Two mainly factors are considered in this phrase, including the task precedence relationship and computational complexity. We design a heuristic algorithm based on the HEFT, which allocates the task to the server with the earliest completion time. The pseudo code of the task allocation is given in Algorithm 2.

Algorithm 2 The task allocation process

Input: DCT and TCT.
Output: makespan of the whole workflow.
1: $p(\pi|X) \leftarrow$ the P-Network
2: $Rank(\pi) \leftarrow \xi - greed$ or greed policy
3: **for** $i \leftarrow 1$ to m **do**
4: **for** $j \leftarrow 1$ to n **do**
5: calculate the start time of $Rank(i)$ in server v_j, $SET(Rank(i), v_j)$;
6: **for** $l \leftarrow 1$ to i **do**
7: **if** $Rank(l)$ is the father node of $Rank(i)$ **then**
8: $time_1 \leftarrow \max_{l\in(1,i),h\in(1,n)}(FET(Rand(l), v_h) + DCT(Rand(l), Rand(i))$
9: **end if**
10: **end for**
11: $time_2 \leftarrow$ the finish time of last task on server v_j
12: $SET(Rank(i), v_j) \leftarrow max(time_1, time_2)$
13: $v_k \leftarrow argmin(SET(Rank(i), v_j))$
14: $v_k \leftarrow Rank(i)$
15: $FET(Rank(i), v_K) \leftarrow SET(Rank(i), v_K) + TCT(Rank(i), v_K)$
16: **end for**
17: **end for**
18: $makespan \leftarrow selectthemostvalueofFETinalltasks$
19: return makespan

In the Algorithm 2, we can get the distribution by the proposed neural network and the task sort by the *ξ-greed* policy in the training phrase and greed policy in the testing phrase (line 1–2). *Rank* denotes the task sort list, and *Rank*(*i*) denotes the task in the *i*th location of Rank. Task allocation phase can be achieved by line 3–17. The algorithm selects the sever v_j with the earliest start time $SET(t_i, v_j)$ for task t_i according the task order in Rank. The algorithm first finds all father nodes of *Rank*(*i*) among the set of tasks that already were allocated. Then, the start time of the *Rank*(*i*) on server v_k is determined by the finish time of its father nodes *Rank*(*l*), the data communication time *DCT*(*Rank*(*l*), *Rank*(*i*)) between the *Rank*(*i*) and *Rank*(*l*), and the earliest start execution time of the server v_k (line 4–12). The start time and scheduling result of the *Rank*(*i*) by traversing all servers can be achieved (line 13–14). The finish time of *Rank*(*i*) on the server v_k is determined by the start time of the task and the task computation time on the server (line 15). After all tasks are scheduled, makespan can get (line 18).

4.3 Training methods based on deep reinforcement learning

We utilize the P-Network model to generate a policy π distribution with parameters over a different task sort. Then, we use the DRL to train the proposed neural network model.

1. Reward function

   The ultimate goal of training via deep reinforcement learning is to endow the higher probabilities to the task sort which can get shorter makespan. Reward function can reflect the optimization problem and guide the training method to adjust the parameters of neural network. Using the symbol θ to represent all parameters of the proposed neural network, we can define the reward function as follows.

   $$J(\theta|S) = E_{\pi\sim p_\theta(\cdot|X)} - F(\pi|S) \tag{11}$$

   In this work, there are two parts to achieve the workflow scheduling, task prioritizing phase and task allocation phase. Thus, we can compute the reward only after finishing the two phase. S is the state space vector referring to Sect. 3.3, and X is the neural network input points referring to Sect. 4.1. During the training process, we utilize a distributed γ to generate the DAG-based workflow cases. And the total training goal refers to sampling from this distribution, i.e. $J(\theta) = E_{S\sim\gamma}J(\theta|S)$.
2. Parameter optimization via Actor-Critic

   Given the environment state, the ultimate goal of reinforcement learning is to obtain the highest reward, and the best action policy can achieve this goal. In this

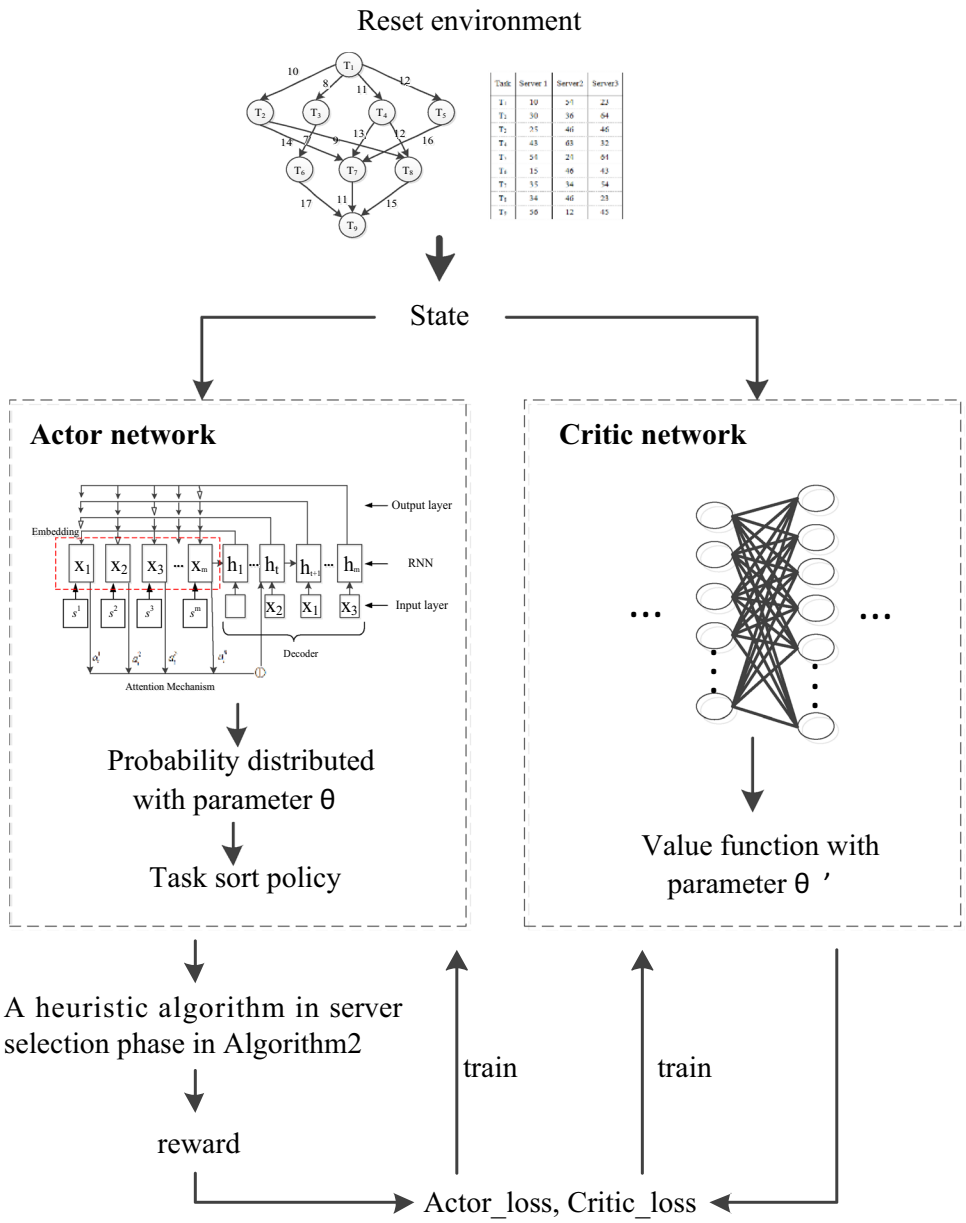


Fig. 6 Flowchat of workflow scheduling based on the deep reinforcement learning

work, action policy is the task sort. We can get the best policy by adjusting the network parameter iteratively. Policy gradient methods are used to train parameters θ in P-Network and θ' in Critic network. The pseudo code of the Actor-Critic architecture training in the workflow scheduling is given in Algorithm 3.

Algorithm 3 Actor-Critic architecture training in the workflow scheduling

Input: characteristics of each task S; the neural network input points X; the sample instances B; training episodes T.
Output: parameter θ and θ'.

Initialize Actor network parameters θ and Critic network parameters θ'.
for $t \leftarrow 1$ to T **do**
  for $i \leftarrow 1$ to B **do**
    $S_i \sim \gamma$
    $p_\theta(\pi|X_i) \leftarrow$ the P-Network
    $p_\theta(\pi|X_i) \leftarrow$ The process of the $Mask$ scheme in Algorithm1
    $\pi_i \leftarrow p_\theta(\pi|X_i)$ according to the $\xi - greed$
    $F(\pi_i|S_i) \leftarrow$ The task allocation process in Algorithm2
  end for
  $d\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}[(F(\pi_i|S_i) - V_{\theta'}(S_i)) \bigtriangledown_{\theta'} logp_\theta(\pi_i|X_i))]$
  $d\theta' \leftarrow \frac{1}{B}\sum_{i=1}^{B}[\bigtriangledown_{\theta'}(F(\pi_i|S_i) - V_{\theta'}(S_i))^2]$
  updating θ and θ'
end for

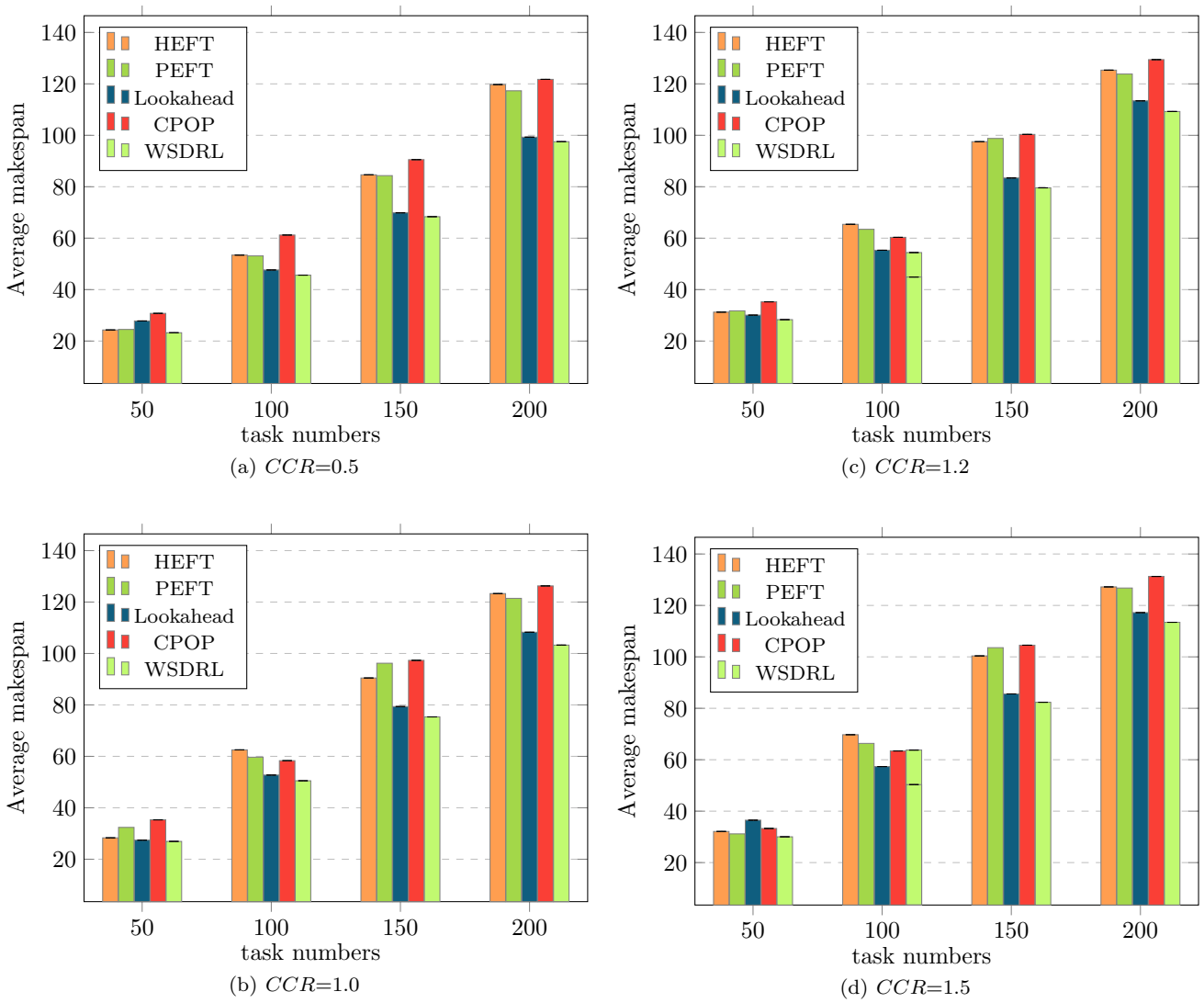


Fig. 7 The average makespan under various number of tasks and CCR

In the Algorithm 3, the DAG-based workflow cases are generated obeying a distributed γ (line 4). Utilizing the proposed P-Network gets the preliminary distribution over different task permutations (line 5), and the ultimate distribution can be predicted by the process of the *Mask* scheme in Algorithm 1 (line 6). According the $p_\theta(\pi|X_i)$, the task sorting policy can be required utilizing the ξ-*greed* (line 7). After training the sample instances using the proposed P-Network (line 3–9), the policy gradient is utilized to update the parameters in the Actor (line 10) and Critic network (line 11). The gradient of the parameters is formulated by the REINFORCE algorithm Ronald (1992), and $V_{\theta'}(S_i)$ is the reward approximation acquiring from the Critic network.

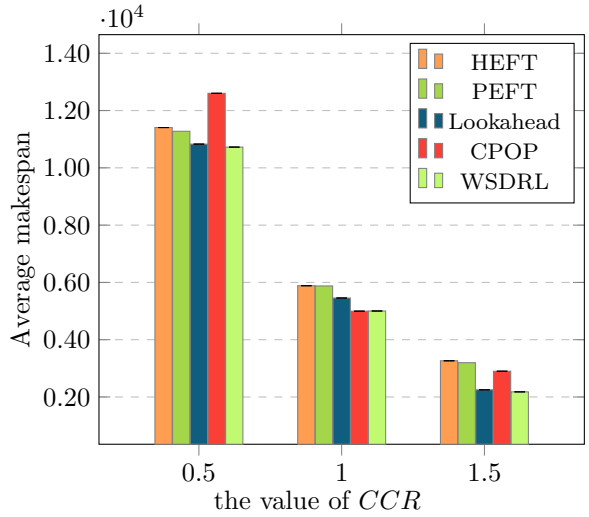
The flowchart of workflow scheduling based on the deep reinforcement learning can be expressed in Fig. 6.

5 Simulations

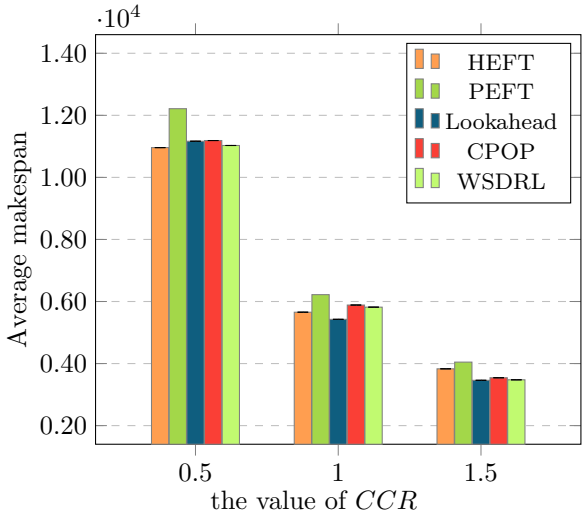
To evaluate the performance of the proposed algorithm, a series of simulations are given. In this section, simulation setup and algorithm performance evaluation are elaborated in detail.

5.1 Simulation setup

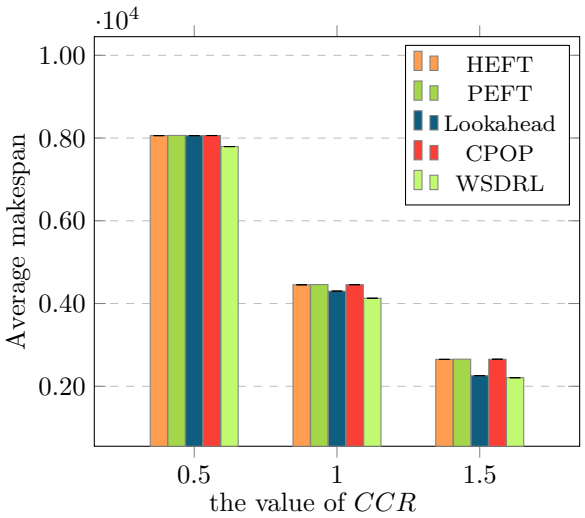
It is complicated and cost-high to evaluate the algorithm performance in the real cloud environment. WorkflowSim is the simulation toolkit extended from CloudSim, which can be utilized to simulate a cloud workflow scheduling environment accurately and used in this work.



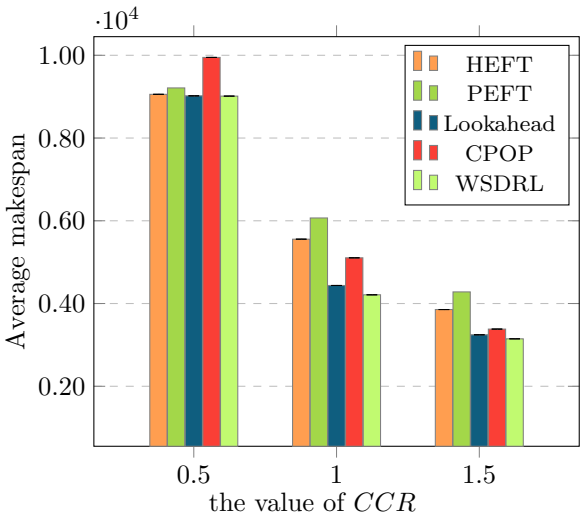
(a) server numbers = 2



(a) server numbers = 10



(b) server numbers = 5



(b) server numbers = 5

Fig. 8 The average makespan of Montage (25 tasks)

Fig. 9 The average makespan of epigenomics (100 tasks)

5.1.1 Related parameters

Two types of workflows are used to evaluate the proposed algorithm, including randomly generated workflows in training and test phrase and workflows of real-word applications in test phrase.

A workflow is described in terms of the workflow structure, which can be described by the amount of tasks and the dependencies between tasks. For randomly generated workflows, given the number of tasks, the dependencies between tasks are randomly generated. In this work, the *DCT* among tasks randomly takes the value from 0 to 10. The communication-to-computation ratio (*CCR*) is utilized to get the average *TCT*, and the *TCT* obeys the normal distribution, the standard deviation of which is 3. For real-world workflows, two types of widely scientific workflow structure are used, including Montage (25 tasks) and epigenomics (100 tasks).

A workflow is regarded as a sample. In each training process, we set 10,000, 25,000 and 50,000 train samples for the batch size of 256, and 2000 test samples. RNN with 128 hidden size is utilized in the P-Network and Critic network. The Adam Optimizer is utilized to train the neural model, and the learning rate is 10^{-4}. The ξ value in the ξ-*greed* policy is significant for the algorithm optimization. The initial ξ value is 0.1, then is updated from 0.1 to 0.05 with the algorithm iteration.

5.2 Performance evaluation

The average makespan are utilized to evaluate the performance of WSDRL. And HEFT, PEFT, Lookahead and CPOP are utilized to make a comparison with the proposed algorithm in this work.

5.2.1 The makespan evaluation of randomly generated workflows

CCR is utilized to reflect the intensive degree of a workflow. The value of *CCR* is large when the communication between tasks is intensive, and the value of *CCR* is small when the computation is intensive. By controlling the value of *CCR*, different types of workflows are generated for the algorithm evaluation. When the value of *CCR* is set to 0.5, 1, 1.2 and 1.5, the average makespan can be obtained under various amounts of tasks by five algorithms using 2000 test samples, which can be described in Fig. 7. In the experiment, 5 servers are utilized to execute those workflows.

In the Fig. 7, the average makespan is the shortest using our proposed algorithm under different amounts of tasks and value of CCR. In addition, Lookahead shows the better performance than other algorithms excepting the proposed algorithm. And CPOP shows the worst performance among the five algorithms. With the increasing amounts of tasks, the performance superiority of our algorithm becomes more obvious because of the disadvantage of the local solution for heuristic algorithms.

5.2.2 The makespan evaluation of real-world workflows

The two kinds of workflows are utilized to calculate the average makespan using different amounts of servers. In each workflow, values of *CCR* are 0.5, 1 and 1.5.

The average makespan of Montage (25 tasks) can be acquired by 2 severs in Fig. 8a and 5 servers in Fig. 8b. Firstly, it can be found that the average makespan reduces with the increasing number of servers. However, the downward trend is not large with the increasing value of *CCR* , that is because it is unwise to allocate tasks to multiple servers when *DCT* is large. Then, WSDRL can acquire a better result.

The average makespan of Epigenomics (100 tasks) can be acquired by 10 severs in Fig. 9a and 20 servers in Fig. 9b. It can be found that WSDRL can achieve the nearly same effect as Lookahead. However, the computational complexity is far lower than Lookahead, because we only need to consider the task allocation phrase after training.

6 Conclusion and future work

A new workflow scheduling algorithm based on the Actor-Critic architecture is proposed to achieve the makespan minimization. The designed architecture combines the neural network model and the heuristic algorithm based on the HEFT to achieve the task sequence and task allocation, considering the solution quality and computational complexity. The algorithm performance evaluation is in aspect of the average makespan, and the simulation result comparing with other four heuristic algorithms shows the effectiveness of the proposed algorithm. In the future work, we will design the algorithm to solve the multi-objective workflow scheduling (Wang et al. 2020) on the basis of this work. Besides, dynamic uncertainties have an important impact on the workflow scheduling, and it is the major work to add dynamic uncertainties in the cloud environment into the design of the algorithm.

Acknowledgements This paper is supported by Humanity and Social Science Research of Ministry of Education (20YJCZH200), Beijing Intelligent Logistics System Collaborative Innovation Center Open Topic (No. BILSCIC-2019KF- 05), Grass-roots Academic Team Building Project of Beijing Wuzi University (No. 2019XJJCTD04).

References

Arabnejad H, Barbosa JG (2014) List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parallel Distrib Syst 25(3):682–694

Asghari A, Sohrabi MK, Yaghmaee F (2020) Online scheduling of dependent tasks of cloud's workflows to enhance resource utilization and reduce the makespan using multiple reinforcement learning-based agents. Soft Comput. https://doi.org/10.1007/s00500-020-04931-7

Barto AG, Sutton RD, Anderson CW (1970) Neuron like elements that can solve difficult learning control problems. IEEE Trans Syst Man Cybern 13(5):834–846

Bittencourt LF, Sakellariou R, Madeira ERM (2010) DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: 2010 18th Euromicro conference on parallel, distributed and network-based processing, pp 27–34. https://doi.org/10.1109/PDP.2010.56

Bodrow W (2017) Impact of industry 4.0 in service oriented firm. Adv Manuf 5(4):394–400

Cai XJ, Hu ZM, Chen JJ (2020a) A many-objective optimization recommendation algorithm based on knowledge mining. Inf Sci 537:148–161

Cai XJ, Hu ZM, Chen JJ (2020b) A many-objective optimization recommendation algorithm based on knowledge mining. Inf Sci 537:148–161

Cai X, Geng S, Wu D, Cai J, Chen J (2020c) A multi-cloud model based many-objective intelligent algorithm for efficient task scheduling in internet of things. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2020.3040019

Cui ZH, Xue F, Cai XJ, Cao Y, Wang GG, Chen JJ (2018) Detection of malicious code variants based on deep learning. IEEE Trans Ind Inform 14(7):3187–3196

Cui ZH, Zhang JJ, Wu D, Cai XJ, Wang H, Zhang WS, Chen JJ (2020) Hybrid many-objective particle swarm optimization algorithm for green coal production problem. Inf Sci 518:256–271

Cui ZH, Xu XH, Xue F, Cai XJ, Cao Y, Zhang WS, Chen JJ (2020) Personalized recommendation system based on collaborative filtering for IoT scenarios. IEEE Trans Serv Comput 13(4):685–695

Dong TT, Xue F, Xiao CB, Li JT (2020) Task scheduling based on deep reinforcement learning in a cloud manufacturing environment. Concurr Comput Pract Exp 32(11):e5654

Dzmitry B, Kyunghyun C, Yoshua B (2015) Neural machine translation by jointly learning to align and translate. In: International conference on learning representations. arXiv:1409.0473

Faragardi HR, Sedghpour MRS, Faziahmadi S, Fahringer T, Rasouli N (2020) GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds. IEEE Trans Parallel Distrib Syst 31(6):1239–1254

Hassan M, Rehmani MH, Chen JJ (2020a) DEAL: differentially private auction for blockchain based microgrids energy trading. IEEE Trans Serv Comput 13(2):263–275

Hassan M, Rehmani MH, Chen JJ (2020b) Differential privacy techniques for cyber physical systems: a survey. IEEE Commun Surv Tutor 22(1):746–789

Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biol Cybern 52(3):141–152

Irwan B, Hieu P, Quoc V L, Mohammad N, Samy B (2016) Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940

Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2013) Characterizing and profiling scientific workflows. Future Gener Comput Syst 29(3):682–692

Karpagam M, Geetha K, Rajan C (2020) A reactive search optimization algorithm for scientific workflow scheduling using clustering techniques. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-020-02480-3

Khan SQ, Ghani A, Khurram M (2020) Frequency-dependent synaptic plasticity model for neurocomputing applications. Int J Bioinspired Comput 16(1):56–66

Kumar H, Giri S (2020) Optimisation of makespan of a flow shop problem using multi layer neural network. Int J Comput Sci Math 11(2):107–122

Liu QM (2020) Integrated deteriorating maintenance and patient scheduling for single medical device with heuristic algorithm. Int J Bioinspired Comput 16(2):121–131

Market Share: IT Services, Worldwide (2019). https://www.gartner.com/en/documents/3983385/market-share-it-services-worldwide-2019. id: g00717813

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533

Mohammadreza N, Afshin O, Martin T, Lawrence VS (2018) Reinforcement learning for solving the vehicle routing problem. arXiv:1802.04240

Nedjah N, Mourelle LD, Morais RG (2020) Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation: a survey—part I. Int J Bioinspired Comput 15(4):207–223

Oriol V, Meire F, Navdeep J (2015) Pointer networks. In: Advances in neural information processing systems, pp 2692–2700. arXiv:1506.03134

Rajasekar P, Palanichamy Y (2020) Scheduling multiple scientific workflows using containers on IaaS cloud. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-020-02483-0

Ronald W (1992) Simple statistical gradient following algorithms for connectionnist reinforcement learning. Mach Learn 8(3–4):229–256

Sun D, Gao S, Liu XY, Li FY, Buyya R (2020) Performance-aware deployment of streaming applications in distributed stream computing systems. Int J Bioinspired Comput 15(1):52–62

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge

Tong Z, Chen HJ, Deng XM, Li KL, Li KQ (2019) A scheduling scheme in the cloud computing environment using deep Q-learning. Inf Sci 512:1170–1191

Tong Z, Deng XM, Chen HJ, Mei J, Liu H (2020) QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. Neural Comput Appl 32(10):5553–5570

Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274

Wang PH, Huang JR, Cui ZH, Xie LP, Chen JJ (2020) A Gaussian error correction multi-objective positioning model with NSGA-II. Concurr Comput Pract Exp 32(5):e5464

Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292

Yuan D, Yang Y, Liu X, Chen JJ (2010) A data placement strategy in scientific cloud workflows. Future Gener Comput Syst 26(8):1200–1214

Zhang X, Li XT, Yin MH (2020) An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem. Int J Bioinspired Comput 15(2):113–124