International Journal of Bio-Inspired Computation

ISSN online: 1758-0374 - ISSN print: 1758-0366
https://www.inderscience.com/ijbic

A hybrid algorithm for workflow scheduling in cloud environment

Tingting Dong, Li Zhou, Lei Chen, Yanxing Song, Hengliang Tang, Huilin Qin

DOI: 10.1504/IJBIC.2023.10055212

Article History:

Received:	16 October 2021
Last revised:	10 March 2022
Accepted:	11 March 2022
Published online:	04 April 2023

A hybrid algorithm for workflow scheduling in cloud environment

Tingting Dong*

School of Information,
Beijing Wuzi University,
Beijing, 101149, China
and
Faculty of Information Technology,
Beijing University of Technology,
Beijing, 100000, China
Email: dongtingting2019@163.com
*Corresponding author

Li Zhou, Lei Chen, Yanxing Song, Hengliang Tang and Huilin Qin

School of Information,
Beijing Wuzi University,
Beijing, 101149, China
Email: zhoulibit@126.com
Email: susychenlei@163.com
Email: yanxing7090091@163.com
Email: tanghengliangbwu@163.com
Email: qhlin9309@163.com

Abstract: The advances in cloud computing promote the problem in processing speed. Computing resources in cloud play a vital role in solving user demands, which can be regarded as workflows. Efficient workflow scheduling is a challenge in reducing the task execution time and cost. In recent years, deep reinforcement learning algorithm has been used to solve various combinatorial optimisation problems. However, the trained models often have volatility and can not be applied in real situation. In addition, evolutionary algorithm with a complete framework is a popular method to tackle the scheduling problem. But, it has a poor convergence speed. In this paper, we propose a hybrid algorithm to address the workflow scheduling problem, which combines deep reinforcement algorithm and evolutionary algorithm. The solutions generated by deep reinforcement learning are the initial population in the evolutionary algorithm. Results show that the proposed algorithm is effective.

Keywords: deep reinforcement learning; evolutionary algorithm; workflow scheduling; multi-objective optimisation.

Reference to this paper should be made as follows: Dong, T., Zhou, L., Chen, L., Song, Y., Tang, H. and Qin, H. (2023) 'A hybrid algorithm for workflow scheduling in cloud environment', *Int. J. Bio-Inspired Computation*, Vol. 21, No. 1, pp.48–56.

Biographical notes: Tingting Dong is currently working toward his PhD at the College of Computer Science, Beijing University of Technology, Beijing, China. Her main research interests include cloud computing, combination optimisation and reinforcement learning.

Li Zhou is a Professor at the School of Information, Beijing Wuzi University, China. She received her PhD from the Beijing Institute of Technology in 2006. Her research interests include computational complexity theory, intelligent optimisation, intelligent logistics and logistics management.

Lei Chen is an Associate Professor at the School of Information, Beijing Wuzi University, China. She received her PhD from the China Agricultural University in 2016. Her research interests include evolutionary algorithm and information system.

Yanxing Song is an Associate Professor at the School of Information, Beijing Wuzi University, China. She received her PhD from the Harbin Institute of Technology in 2010. Her research interests include operational research optimisation and evolutionary algorithm.

Hengliang Tang is a Professor at the School of Information, Beijing Wuzi University, China. He received his PhD from the Beijing University of Technology in 2011. His research interests include evolutionary algorithm and deep reinforcement learning.

Huilin Qin is an Associate Professor at the School of Information, Beijing Wuzi University, China. Her research interests include evolutionary algorithm and deep reinforcement learning.

1 Introduction

With the progress in science and technology, people's demand for computing and storage resources is growing rapidly. And, cloud computing can achieve their demand. Large amount of computing resources are deployed in the cloud computing centre to be provided to users on demand, so as to realise resource sharing (Jia et al., 2021; Wang et al., 2021). Among them, the user's request is regarded as a workflow, which can be abstracted as a directed acyclic graph (DAG). A workflow is composed of a group of task nodes with dependency relationship, that is, the child task must receive all the data passed by the parent task before it can start execution. Workflow scheduling is to map tasks in the workflow to appropriate servers under certain goals and constraints (Versluis and Iosup, 2021), which is an NP-hard problem, and it is difficult to achieve optimal scheduling.

Heuristic algorithms (Sahni and Vidyarthi, 2018; Djigal et al., 2021; Faragardi et al., 2020) are often used to solve the workflow scheduling problem, which is to design heuristic rules according to specific problem scenarios. The logic of heuristic algorithms is simple and easy to understand, and the calculation speed is fast. Common heuristic algorithms include list-based scheduling, task replication methods, and cluster-based scheduling (Verma and Kaushal, 2015). Among them, Topcuoglu et al. (2002) proposed heterogeneous earliest-finish-time (HEFT) algorithm, which is one of the most popular list_based algorithm. However, the design of a high-performance heuristic algorithm is very difficult and requires a lot of trial and error and expert knowledge. In addition, most heuristic algorithms can only obtain local optimal solutions, and when the problem scenario (such as problems, goals, constraints, etc.) changes, the heuristic rules need to be redesigned. Evolutionary algorithm (EA) (Zhang et al., 2022; Cai et al., 2021; Muteeh, 2021; Konda et al., 2021) is another popular method to solve the workflow scheduling problem, such as genetic algorithm (GA), particle swarm algorithm (PSO), ant colony algorithm. EA (Cui et al., 2021a, 2021b; Zhang et al., 2021) has a unified algorithm frameworks, and improves the quality of solutions through continuous iterative search. EA does not require the construction of complex mathematical models. Many problems in different scenarios can be solved by adjusting the fitness function, and global optimisation can be achieved by iteration. In particular, it is more suitable for multi-objective problems.

Casas et al. (2018) proposed a GA-ETI algorithm to address the workflow scheduling in cloud environment, considering the task execution time and monetary cost. For the order assignment and supplier selection and production line scheduling in the cloud manufacturing environment, Laili et al. (2020) used six multi-objective EA to integrated scheduling. However, the initial population of most EAs is generated randomly. Although this can increase the diversity of the population, a poor initial population will reduce the optimisation and convergence speed of the algorithm. Aziza (2020) proposed a HEFT-GA algorithm to solve the workflow scheduling in cloud environment, which is a hybrid algorithm that the initial population of GA consists of an individual generated by HEFT and the rest individuals generated randomly. However, for the same problem, HEFT can only produce a single solution, which will reduce the diversity of the population. Alipour (2018) proposed a hybrid algorithm to solve the traveling salesman problem, which combined multiagent reinforcement learning (RL) and GA. In this algorithm, the initial solutions in GA are generated by Q-learning, which is a kind of RL.

Recently, with the continuous development of machine learning (ML), more and more research has been proposed to solve the scheduling problem using learning-based approaches (Bengio et al., 2021). RL (Sutton, 1998) is more suitable than deep learning (DL) for solving scheduling problems, because instead of learning a model from a training set with labels, RL learns that from the feedback information of the environment. This is crucial for solving workflow scheduling problems. Because the workflow scheduling problem is an NP-hard problem, it is not easy or impossible to generate labels for high-quality scheduling results. Ding et al. (2020) proposed a QEEC method based on Q_learning framework and queuing model to address the task assignment and scheduling considering energy consumption in cloud environment. Guo et al. (2021) proposed a DeepRM_Plus method to address the resource management problem, which used deep RL and used the imitate learning to speed up the model convergence. However, they can not consider the task dependency. Tong et al. (2020) proposed a DQTS method to address the task scheduling in cloud environment, considering the makespan and load balance. In this algorithm, a deep Q_learning framework was used.

In this paper, a hybrid algorithm combining deep RL and EA is designed to solve the multi-objective workflow scheduling problem. The main contributions of this paper are:

1 A workflow scheduling model is built aiming to minimise the completion time of the workflow and

the execution cost, and the dependency relationship between tasks is as the constraint.

2 In order to accelerate the convergence speed and improve the algorithm optimisation, the workflow scheduling solution generated by DQN is used as the initial population of the EA, and NSGA-II is taken as an example to introduce the hybird algorithm.

3 To verify the effectiveness of the algorithm, we use a variety of real-world scientific workflows for simulation experiments. Our proposed algorithm make a comparison with the random initial population and HEFT-based initial population on multiple EAs. The simulation results show that our proposed algorithm is superior to the other two methods in terms of the task execution time and cost.

The rest of this paper is organised as follows. In Section 2, the relevant theories of DRL and EAs are introduced. In Section 3, a model of workflow scheduling is built. In Section 4, we propose a hybrid algorithm for workflow scheduling. Section 5 presents simulation experiments and analyses the results. Section 6 gives the conclusions of this paper and proposes future work.

2 Preliminaries

2.1 Deep RL

Workflow scheduling problem is a sequential decision problem that maps tasks to appropriate servers. It can be regarded as Markov decision process (MDP), and a decision model is built. RL is based on the MDP framework. At each decision step, the agent chooses an action in the current state and gets a reward value, which is used to evaluate the quality of the executed action. After the action is executed, the agent enters the next state. Through the continuous interaction between the agent and the environment, and the continuous trial and error and learning process from the feedback information, a series of optimisation strategies are finally obtained.

The MDP is mainly composed of four important parts (S, A, P, R), of which:

1 S: State space. S is mainly used to describe a set of environmental states.

2 A: Action space. A is the set of all actions that can be taken.

3 P: State transition probability. P is the probability of transition from the current state to the next state.

4 R: Reward function. R is the evaluation value of the action in the process of taking an action from the current state to the next state.

When the state space of the problem is not very large, the Q value can be stored by constructing a state-action pair, and using RL based on Q-learning or SARSA to solve the optimal scheduling problem (Watkins and Dayan, 1992). Among them, the typical Q value update formula of Q-learning_based RL is as follows:

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[r_k + \gamma \max_a Q(s_{k+1}, a_k) - Q(s_k, a_k) \right] \quad (1)$$

where γ is the discount factor, and α is the learning rate, and $r_k + \gamma \max_a Q(s_{k+1}, a_k)$ is defined as the target value, and $r_k + \gamma \max_a Q(s_{k+1}, a_k) - Q(s_k, a_k)$ is the deviation between the target value and the Q value at decision step k.

The advantage of the above algorithm is that it can guarantee convergence in a finite decision step. The disadvantage is that, on the one hand, it is not suitable for large-scale scheduling problems. Because the state space of large-scale scheduling problems is huge, the Q value table will be too large, which is not conducive to storage. On the other hand, when the state space is continuous, the continuous space needs to be discretised, which will lead to the omission of some important state information.

To overcome the above problems, a neural network is used to generate an approximate function with parameters instead of the Q value table. Through continuous training of network parameters, the value function is optimised to obtain the optimal scheduling strategy. Among them, the typical representative is the DQN algorithm proposed by Google DeepMind in 2015 with the background of game training (Mnih et al., 2015). In addition, the algorithm breaks the correlation between data from the following two aspects:

1 Experience replay pool: During the training process in the neural network, the training datas must be independent and identically distributed. The datas from the sequential decision-making are correlated, which will cause the neural network to be unstable. Experience replay pool is to store the data of each learning in the experience pool. When the data in the experience pool reaches a certain amount, the uniform random sampling method is used to extract a certain amount of data from the experience pool for network training.

2 The target network: Traditional DRL is a simple combination of DL and RL. DRL designs a neural network to generate a Q value with a parameter ω instead of the Q value table, and uses the idea of constructing deviation in Q-learning to construct a loss function. Finally, use the gradient descent method to train the parameters. The loss function can be expressed as:

$$L_1(\omega) = E(y_k - Q(s_k, a_k; \omega))^2 \quad (2)$$

$$y_k = r_k + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}; \omega) \quad (3)$$

Among them, y_i is called the target value function. Because the two parameters in formulas (2) and (3)

are the same, it will cause the two value functions to be correlated, which will lead to premature convergence of the training and make the neural model unstable.

For the above problem, DQN is to set a neural network to generate a Q value with parameter ω (the neural network is called an evaluation network). In addition, DQN also designs a neural network to calculate the target value function with parameter ω^- (the neural network is called the target network). The structure of the two networks is the same, but the parameters are different. The parameter in the evaluation network is updated every decision step, and is copied to the target network every fixed decision steps. This helps to eliminate the correlation between the two value functions and makes the trained model more stable. The loss function is expressed as:

$$L_2(\omega) = E(y_k - Q(s_k, a_k; \omega))^2 \tag{4}$$

$$y_k = r_k + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}; \omega^-) \tag{5}$$

3 Problem formulation

In the cloud data centre, a set of servers are deployed, and can be represented by Q, $Q = \{q_1, q_2, ..., q_m\}$. A workflow is executed in a cloud data centre, thus, the average bandwidth can be used to represent the bandwidth between each two servers. The number of CPUs and memory size of each server are different, thus, the computing capacity for each server is different. We use the CU to represent the computing capacity in this paper. On-demand provisioning is adopted in the cloud environment, that is, users only pay the provider according to their required resources and time slots.

A workflow abstracted as a DAG is comprised of a set of edges D and tasks T. A set of tasks is defined by $T = \{t_1, t_2, ..., t_n\}$, and a set of edges is defined by $D = \{d_{ij} | i \neq j, i, j \in \{1, 2, ..., n\}\}$, where d_{ij} represents the output data size from task t_i to t_j. If d_{ij} is zero, then the two tasks have no dependency relationship. Otherwise, task t_i is the father task of task t_j, and t_j is the son task of task t_i. The task without any father task is defined as t_enter, and the task without any son task is defined as t_entry. A workflow with ten tasks is shown in Figure 1.

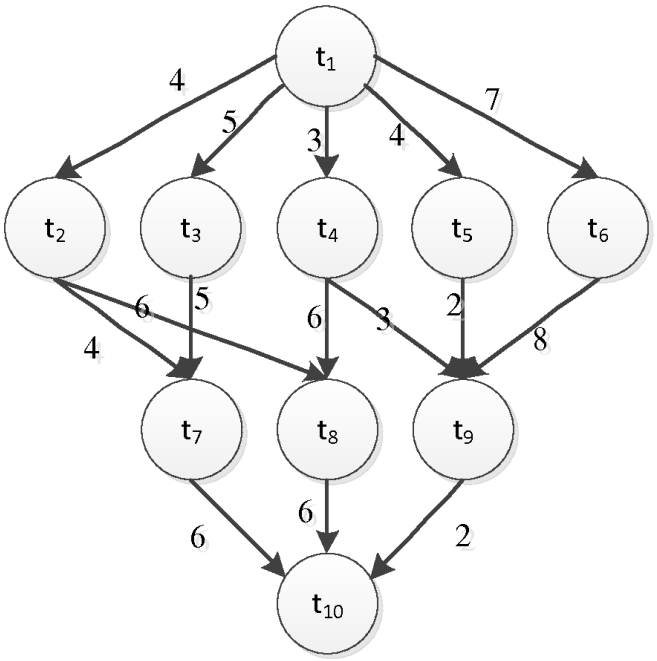
In this paper, workflow scheduling is to map all tasks in a workflow to servers in the cloud environment, aiming to minimise the task execution time and the resource operation cost. However, they are two opposing goals, because the server with a higher computing capacity often has a higher cost. The two goals are defined as follows.

1 Makespan:The task execution time is also called the makespan, which is the finishing time of the last task.

$$makespan = \max_{i \in [1,n]} FT(t_i, q_j) \tag{6}$$

where $FT(t_i, q_j)$ is the finish time of the task t_i on the server q_j.

Figure 1 A workflow with ten tasks



2 Cost: Servers executing all tasks need to speed time, and each server has different charge price. In addition, data transfer also needs to pay. Thus, the cost is defined by

$$cost_1 = \sum_{j=1}^{m} \sum_{i=1}^{n} (FT(t_i, q_j) - ST(t_i, q_j)) x_{ij} p_j \tag{7}$$

$$x_{ij} = \begin{cases} 1, \text{if task } t_i \text{ in server } q_j \\ 0, \text{otherwise} \end{cases} \tag{8}$$

where $ST(t_i, q_j)$ and $FT(t_i, q_j)$ are the start and finish time of the task t_i on the server q_j, and p_j is the price of server q_j.

$$cost_2 = \sum_{i=1}^{n} TM(t_i, t_j) p' \tag{9}$$

$$TM(t_i, t_j) = \begin{cases} 0, & \text{if } t_i \text{ and } t_j \text{ in the same server} \\ \frac{d_{ij}}{bw}, & \text{otherwise} \end{cases} \tag{10}$$

where bw is the average bandwith, and $TM(t_i, t_j)$ is the data transfer time, and p' is the price of data transfer time.

Thus, the cost can be defined by

$$cost = cost_1 + cost_2 \tag{11}$$

4 Algorithm description

4.1 Workflow scheduling based on deep RL

Firstly, we define the MDP of workflow scheduling, including state space, action space, reward and state transition process.

1 State space: At each decision step, the state of workflow scheduling is defined as the input of neural network. And, the definition of state space can reflect the execution of tasks and the state of servers. At the current time, each server load and the the load standard deviation are included in the state space.

2 Action space: For each task, we need to determine which server is used to execute it. Thus, all servers consist of action space. To balance the exploitation and exploration, ξ-greed is used to select actions.

3 Reward: The goal is to minimise the makespan and reduce the cost, and reward function should reflect to the two goals. Two rewards are defined by

$$Reward_1 = \frac{makespan(s_k) - makespan(s_{k+1})}{makespan(s_{k+1})} \quad (12)$$

$$Reward_2 = \frac{cost(s_k) - cost(s_{k+1})}{cost(s_{k+1})} \quad (13)$$

$$Reward = Reward_1 + Reward_2 \quad (14)$$

where s_k is the state at decison step k, and s_{k+1} is the state at decision step $k+1$.

4 State transition process: For a workflow, first, we use the updating ranking (Topcuoglu et al., 2002) to generate a task list. And, at the current state, a task is selected in order of the task list. Then, an action is determined for this task. We can obtain the reward after executing this action, and the environment information is updated to the next state. An episode refers that all tasks in a workflow are allocated.

5 The parameter training process

Based on Subsection 2.1, the pseudo-code of the workflow scheduling based on DQN is as Algorithm 1.

4.2 Workflow scheduling based on NSGA-II

For the solution of MOPs, as early as 1994, Srinivas and Deb (1994) proposed the non-dominated sorting genetic algorithm (NSGA). Compared with GA, NSGA adopts the strategy of non-dominated stratification and fitness sharing, which can give good individuals a better chance to reach the next generation, and obtain a more evenly distributed pareto optimal solution set. However, NSGA also has some disadvantages:

1 During the non-dominated sorting, multiple searches are generally required, and the number of searches will increase with the increasing number of goals (l) and the size of the population (N). This results in a relatively high time complexity $O(lN^3)$;

2 Lack of elite strategy. The elite strategy can retain good individuals and accelerate the convergence of the Pareto front.

3 You need to specify special sharing parameters yourself, which have a relatively large impact on the diversity of the population.

Algorithm 1 Workflow scheduling based on DQN

Input: Episode number Num; Minebatch size B; Replay memeory size M
Output: Parameter ω and ω^-
Initialise state space
Ep = 0
while $Ep < Num$ **do**
  task list $\leftarrow$ updating ranking
  Ep += 1
  for t_i *in task list* **do**
    $a_k = \begin{cases} \arg\max_a Q(s_k, a; \omega), & probability \in [0, \xi] \\ randomly, & probability \in (\xi, 1] \end{cases}$
    next state s_{k+1} and reward r_k
    store $< s_k, a_k, r_k, s_{k+1} >$
    sample $< s_k, a_k, r_k, s_{k+1} >$ with B
    if t_i *is last task* **then**
      $y_k = r_k$
    end
    else
      $y_k = r_k + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}; \omega^-)$
    end
    $L(\omega) = E(y_k - Q(s_k, a_k; \omega))^2$
    update $\omega \leftarrow$ *gradient descent*
    update $\omega^- \leftarrow \omega$ *(every h step)*
  end
end
Return(ω and ω^-)

Therefore, in 2002, Deb et al. (2002) improved NSGA and proposed a NSGA with elite strategy (NSGA-II). Compared with NSGA, this algorithm has the following advantages:

1 It proposes a fast non-dominated sorting algorithm, which reduces the time complexity to $O(lN^2)$.

2 It joins the elite strategy to ensure that the best individuals will not be lost. This improves the overall level of the population.

3 It adds a crowding operator and does not need to determine a shared parameter. This ensures the diversity of the population.

This paper takes the NSGA-II algorithm as an example and describes its combination with DQN to solve the workflow scheduling problem.

For the workflow scheduling, a chromosome (or an individual) can be represented by a complete workflow scheduling solution. And, the gene in the chromosome indicates a task. The value on the gene is the VM number. A population consists of multiple chromosomes, that is, a

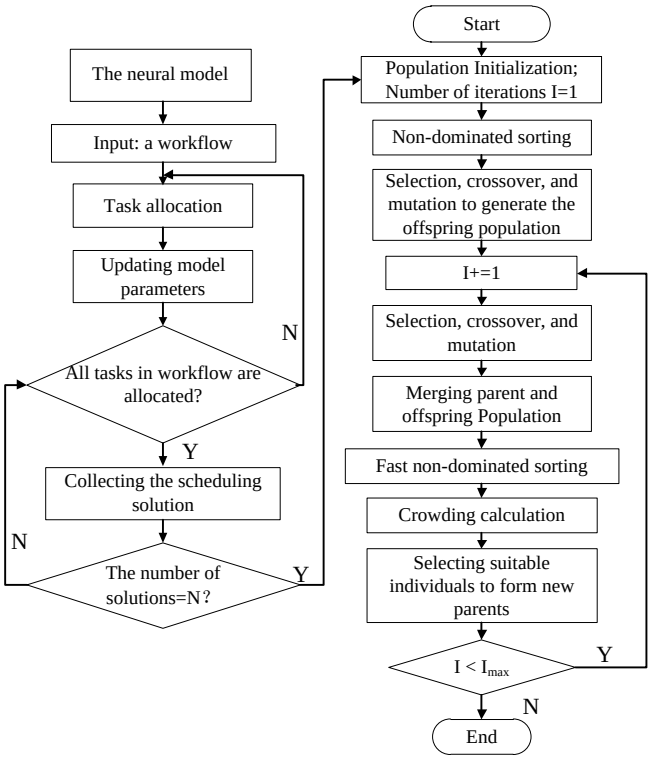
population is comprised of multiple workflow scheduling solution.

The basic idea of NSGA-II is as follows.

Firstly, the initial population of scale N is generated randomly, and the first generation offspring population is obtained by three basic operations of GA: selection, crossover and mutation after non-dominated sorting; secondly, starting from the second generation, the parent population and the offspring population are merged for fast non-dominated sorting, and at the same time, the crowding degree of individuals in each non-dominated layer is calculated, and suitable individuals are selected according to the non-dominated relationship and the crowding degree of individuals forms a new parent population; finally, a new offspring population is generated through the basic operation of GA; and so on, until the conditions for ending the program are met.

The different between the NSGA-II and the proposed algorithm in this paper is about the initial population generation. In this paper, all workflow scheduling solutions in the initial population are generated by DQN. Firstly, we train the neural network for a fixed episodes and save the model. Then, for a special workflow, we invoke the saved model and iterate N episode, which can generate N solutions. During this process, the parameter of neural network is still changed at each episode, which can generate various solutions. The flowchart of the proposed hybrid algorithm is in Figure 2.

Figure 2 The flowchart of the proposed hybrid algorithm



4.3 Time complexity analysis

For the NSGA-II with a randomly generated initial population, the time complexity is mainly from the fast non-dominated sorting. And, the time complexity is $O(lN^2)$, where l is the number of goals and N is the size of population. If the initial population of NSGA-II is generated by DQN, the time complexity is still $O(lN^2)$. Firstly, for a special workflow, DQN can generate a solution through n iterations. Thus, the time complexity of the initial population generated by DQN is $O(nN)$, which is far less than $O(lN^2)$. To sum up, the NSGA-II with the initial population generated by DQN does not increase the computing time cost.

5 Simulation evaluation

5.1 Experimental setting

5.1.1 Workflow and server parameter setting

Synthetic workflows from the real-world scientific applications are utilised to evaluate the algorithm performance, which are published by Pegasus project. In this work, three workflow types are used to be the experiment data, including montage, cybershake, and inspiral. Workflows in a workflow types have the similar structure but different task numbers. To test the algorithm stability, a different number of tasks in each type of workflows are selected. Each workflow is expressed by an XML file, which consists of each task length, the input and output data size of each task, and the DAG structure.

Table 1 The computing capacity and price of each server

Sever	*Computing capacity*	*Price*
q_1	1.7	0.06
q_2	3.75	0.12
q_3	3.75	0.113
q_4	7.5	0.24

Table 2 Parameter setting

Parameters	*Description*	*Value*
γ	Discount factor	0.9
α	Learning rate	1×10^{-6}
ξ	ξ-greed	0.9
M	Replay memory size	1×10^{6}
B	Minebatch size	128
Num	Episode number	10,000
N	Population size	100
I	Iteration	300

Four server types from Amazon EC2 instances are selected, and Table 1 describes the computing capacity and price of each server. The average bandwidth between VMs is set to be 10 MB/S, and the data transfer price (p') is 0.1.

Other parameters about evolutionary and DQN are as Table 2.

Table 3 Makespan of real-world workflow scheduling by $NSGA\_II$

Metric	*Workflows*	*R_NSGA_II*	*H_NSGA_II*	*D_NSGA_II*
Best	$Montage\_50$	2.5680	1.3013	*1.2667*
	$Montage\_100$	2.0347	2.0347	*1.8427*
	$Cybershake\_50$	0.6933	1.3000	*0.5941*
	$Cybershake\_100$	23.6980	*0.6013*	0.8133
	$Inspiral\_50$	101.6600	101.6560	*101.6560*
	$Inspiral\_100$	*37.3910*	39.6480	56.2107
Worst	$Montage\_50$	*37.3240*	50.9090	37.5176
	$Montage\_100$	66.9150	*57.6621*	628.2893
	$Cybershake\_50$	235.4400	271.3570	*238.7641*
	$Cybershake\_100$	496.2000	540.7445	*368.6267*
	$Inspiral\_50$	1,114.7000	746.9784	*737.1471*
	$Inspiral\_100$	*755.6400*	1,522.0233	1,179.1020
Mean	$Montage\_50$	16.7790	*15.7654*	16.8382
	$Montage\_100$	23.1400	*18.8548*	153.5838
	$Cybershake\_50$	238.7640	123.3277	*111.3251*
	$Cybershake\_100$	218.2200	200.6230	*159.1581*
	$Inspiral\_50$	*238.4300*	3,323.2568	276.0462
	$Inspiral\_100$	*298.7600*	366.2463	360.2245

Table 4 Cost of real-world workflow scheduling by $NSGA\_II$

Metric	*Workflows*	*R_NSGA_II*	*H_NSGA_II*	*D_NSGA_II*
Best	$Montage\_50$	54.2600	*53.6223*	54.4554
	$Montage\_100$	113.3400	112.2100	*107.1777*
	$Cybershake\_50$	*317.6100*	322.3612	318.9448
	$Cybershake\_100$	727.3200	726.2047	*698.8867*
	$Inspiral\_50$	1,049.1000	1,048.4322	*1,040.5970*
	$Inspiral\_100$	1,881.7000	1,877.5974	*1,864.5670*
Worst	$Montage\_50$	54.8030	*54.1568*	55.0043
	$Montage\_100$	114.4700	113.338	*108.2518*
	$Cybershake\_50$	*320.7400*	325.5517	322.1190
	$Cybershake\_100$	734.5500	726.2047	*705.9314*
	$Inspiral\_50$	1,059.5000	1,058.8664	*1,050.9814*
	$Inspiral\_100$	1,900.6000	1,896.2901	*1,883.3272*
Mean	$Montage\_50$	54.5320	*53.8899*	54.7296
	$Montage\_100$	113.9100	112.7750	*107.7152*
	$Cybershake\_50$	*319.1800*	323.9500	320.5323
	$Cybershake\_100$	730.9300	729.8180	*702.4213*
	$Inspiral\_50$	1,054.3000	1,053.6500	*1,045.7917*
	$Inspiral\_100$	1,891.2000	1,886.9315	*1,873.9451*

Table 5 Makespan of real-world workflow scheduling by $NSGA\_III$

Metric	*Workflows*	*R_NSGA_III*	*H_NSGA_III*	*D_NSGA_III*
Best	$Montage\_50$	1.3013	1.4320	*1.3013*
	$Montage\_100$	2.0347	2.0347	*1.8427*
	$Cybershake\_50$	16.4570	0.6880	*0.5176*
	$Cybershake\_100$	4.1157	*0.5882*	1.2347
	$Inspiral\_50$	104.9200	*101.6560*	127.3173
	$Inspiral\_100$	91.1710	*68.3973*	120.3657
Worst	$Montage\_50$	57.9900	50.9090	*30.8942*
	$Montage\_100$	*58.9360*	93.7136	628.1010
	$Cybershake\_50$	303.6000	324.7011	*228.9518*
	$Cybershake\_100$	577.3000	537.4717	*347.0301*
	$Inspiral\_50$	877.5600	1,114.1294	*842.0193*
	$Inspiral\_100$	996.8100	*831.1279*	1,091.6282
Mean	$Montage\_50$	*15.4412*	16.7146	18.0710
	$Montage\_100$	*21.0900*	22.5713	205.1678
	$Cybershake\_50$	124.6900	*123.1372*	123.4762
	$Cybershake\_100$	235.8800	194.2094	*168.6120*
	$Inspiral\_50$	*280.4600*	330.3970	407.8018
	$Inspiral\_100$	*332.8100*	356.9950	387.3123

Table 6 Cost of real-world workflow scheduling by $NSGA\_III$

Metric	*Workflows*	*R_NSGA_III*	*H_NSGA_III*	*D_NSGA_III*
Best	$Montage\_50$	54.3300	*53.5938*	54.2049
	$Montage\_100$	113.1500	112.0050	*106.3586*
	$Cybershake\_50$	*317.5400*	318.7685	317.8310
	$Cybershake\_100$	725.4700	722.1321	*697.8720*
	$Inspiral\_50$	1,047.1000	1,047.9112	*1,040.7192*
	$Inspiral\_100$	1,880.2000	1,876.9745	*1,865.0962*
Worst	$Montage\_50$	54.8720	*54.1276*	54.7461
	$Montage\_100$	114.2800	113.1200	*107.4227*
	$Cybershake\_50$	*320.7200*	321.9906	320.9870
	$Cybershake\_100$	732.6800	729.3767	*704.7685*
	$Inspiral\_50$	1,057.6000	1,058.3639	*1,051.0862*
	$Inspiral\_100$	1,899.1000	1,895.8292	*1,883.6782*
Mean	$Montage\_50$	54.6010	*53.8610*	54.4744
	$Montage\_100$	113.7100	112.5622	*106.8907*
	$Cybershake\_50$	*319.1300*	320.3761	319.4048
	$Cybershake\_100$	729.0700	725.7479	*701.3274*
	$Inspiral\_50$	1,052.3000	1,053.1340	*1,045.9039*
	$Inspiral\_100$	1,889.7000	1,886.4045	*1,874.3881*

Table 7 Makespan of real-world workflow scheduling by $RVEA$

Metric	*Workflows*	*R_RVEA*	*H_RVEA*	*D_RVEA*
Best	$Montage\_50$	*1.2667*	2.5680	1.3013
	$Montage\_100$	2.5600	2.8747	*1.8427*
	$Cybershake\_50$	0.7253	0.9235	*0.5824*
	$Cybershake\_100$	*0.4118*	0.6941	0.4941
	$Inspiral\_50$	*63.6590*	63.6590	72.3490
	$Inspiral\_100$	66.9470	68.3970	*34.1990*
Worst	$Montage\_50$	*51.1860*	126.9100	53.7450
	$Montage\_100$	77.1030	*62.5250*	628.1900
	$Cybershake\_50$	526.4800	351.4400	*290.7500*
	$Cybershake\_100$	*567.4300*	597.1100	599.9900
	$Inspiral\_50$	1,256.2000	*846.6700*	896.3100
	$Inspiral\_100$	1,385.5000	*1,076.8000*	1,146.4000
Mean	$Montage\_50$	16.7090	20.7000	*15.3550*
	$Montage\_100$	*21.0200*	21.1580	134.8600
	$Cybershake\_50$	133.4600	114.7400	*109.9700*
	$Cybershake\_100$	231.1800	230.4200	*200.2700*
	$Inspiral\_50$	*248.4200*	307.1800	350.6600
	$Inspiral\_100$	373.6800	347.4900	*299.1100*

Table 8 Cost of real-world workflow scheduling by $RVEA$

Metric	*Workflows*	*R_RVEA*	*H_RVEA*	*D_RVEA*
Best	$Montage\_50$	53.6290	*52.8080*	53.5400
	$Montage\_100$	112.9100	111.3400	*107.3300*
	$Cybershake\_50$	*317.1500*	319.2700	321.5900
	$Cybershake\_100$	715.9900	720.6200	*693.8500*
	$Inspiral\_50$	1,049.1000	1,052.5000	*1,038.2000*
	$Inspiral\_100$	1,876.4000	1,870.4000	*1,861.9000*
Worst	$Montage\_50$	54.1700	*53.3420*	54.0820
	$Montage\_100$	114.0300	112.4400	*108.4100*
	$Cybershake\_50$	*320.3400*	322.4600	324.8000
	$Cybershake\_100$	723.1600	727.7800	*700.7700*
	$Inspiral\_50$	1,059.7000	1,063.0000	*1,048.6000*
	$Inspiral\_100$	1,895.1000	1,889.2000	*1,880.5000*
Mean	$Montage\_50$	53.8990	*53.0750*	53.8110
	$Montage\_100$	113.4700	111.8900	*107.8700*
	$Cybershake\_50$	*318.7500*	320.8700	323.1900
	$Cybershake\_100$	719.5900	724.2100	*697.3100*
	$Inspiral\_50$	1,054.4000	1,057.8000	*1,043.4000*
	$Inspiral\_100$	1,885.8000	1,879.8000	*1,871.2000*

5.1.2 Baseline algorithms

Three EAs are selected, including NSGA-II, NSGA-III and RVEA. And, the two generation methods of the initial population are used for comparative experiments. The first one is that the initial population is generated randomly. The second one is that the initial population consists of an individuals generated by HEFT and the rest individuals generated randomly. And, the different generation methods of the initial population and EA are combined to produce a new method. For example, R_NSGA_II represents the generation method of the initial population in $NSGA\_II$ is random. H_NSGA_II represents the generation method of the initial population in $NSGA\_II$ is HEFT and random. And, D_NSGA_II represents the generation method of the initial population in $NSGA\_II$ is DQN.

5.2 Simulation results and analysis

Under different real-world workflows, Tables 3 and 4 show the makespans and cost using R_NSGA_II, H_NSGA_II and D_NSGA_II. Tables 5 and 6 show the makespans and cost using R_NSGA_III, H_NSGA_III and D_NSGA_III. Tables 7 and 8 show the makespans and cost using R_RVEA_III, H_RVEA_III and D_RVEA_III. For each goals, we can obtain the maximum (best), average (mean), and minimum (worst) value for each workflow by three algorithms. Among them, the best result is highlighted by black.

For the same EA, the results show that our proposed algorithm can obtain better values in terms of makesppan and cost. Sometimes, the results by our algorithm are not the best. That is because the two goals are contradictory, and it is necessary to sacrifice one for another. Overall, for the generation methods of three initial populations, the performance of the proposed algorithm in this paper is better than the other two algorithms.

6 Conclusions

In this paper, a hybrid algorithm is proposed to address the workflow scheduling problem, aiming to minimise the task execution time and cost. In this algorithm, deep RL and EA are combined, that is, the initial population in EA is generated by deep RL. Three real-world workflow types are used to be the dataset. Different generation methods of initial population are set and combined with different EAs to verify the performance of the algorithm. Results shows that our proposed algorithm is effective.

However, it may be more than two objective for the workflow scheduling in the cloud environment. In the future, we will consider the many-objective workflow scheduling.

Acknowledgements

This work was supported by the Research on Intelligent Inventory Optimisation Decision Driven by Cata (Grant No. 2021XJKY01), Humanity and Social Science Research of Ministry of Education (Grant No. 20YJCZH200), Beijing Intelligent Logistics System Collaborative Innovation Center Open Topic (Grant No. BILSCIC-2019KF-05), National Social Science Project-Research on Logistics Service Quality Improvement and Low Carbon Governance Mechanism (Grant No. 21FGLB046), Beijing Social Science Fund (Grant No. 20GLB026), National Natural Science Foundation of China (Grant No. 72101033), General Program of Science and Technology Development Project of Beijing Municipal Education Commission of China (Nos. KM202110037002, KM202210037004).

References

Alipour, M.M.R.S.F.D.M.E.A. (2018) 'A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem', *Neural Computing and Applications*, Vol. 30, pp.2935–2951.

Aziza, H.K.S. (2020) 'A hybrid genetic algorithm for scientific workflow scheduling in cloud environment', *Neural Computing and Applications*, Vol. 32, pp.15263–15278.

Bengio, Y., Lodi, A. and Prouvost, A. (2021) 'Machine learning for combinatorial optimization: a methodological tour D'horizon', *European Journal of Operational Research*, Vol. 290, No. 2, pp.405–421.

Cai, X., Geng, S., Wu, D., Cai, J. and Chen, J. (2021) 'A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things', *IEEE Internet of Things Journal*, Vol. 8, No. 12, pp.9645–9653.

Casas, I., Taheri, J., Ranjan, R., Wang, L. and Zomaya, A.Y. (2018) 'GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments', *Journal of Computational Science*, Vol. 26, pp.318–331.

Cui, Z., Jing, X., Zhao, P., Zhang, W. and Chen, J. (2021a) 'A new subspace clustering strategy for AI-based data analysis in iot system', *IEEE Internet of Things Journal*, Vol. 8, No. 16, pp.12540–12549.

Cui, Z., Zhao, Y., Cao, Y., Cai, X., Zhang, W. and Chen, J. (2021b) 'Malicious code detection under 5G HetNets based on multi-objective rbm model', *IEEE Network*, Vol. 35, No. 2, pp.82–87.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp.182–197.

Ding, D., Fan, X., Zhao, Y., Kang, K., Yin, Q. and Zeng, J. (2020) 'Q-learning based dynamic task scheduling for energy-efficient cloud computing', *Future Generation Computer Systems*, Vol. 108, pp.361–371.

Djigal, H., Feng, J., Lu, J. and Ge, J. (2021) 'IPPTS: an efficient algorithm for scientific workflow scheduling in heterogeneous computing systems', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 5, pp.1057–1071.

Faragardi, H.R., Saleh Sedghpour, M.R., Fazliahmadi, S., Fahringer, T. and Rasouli, N. (2020) 'GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 31, No. 6, pp.1239–1254.

Guo, W., Tian, W., Ye, Y., Xu, L. and Wu, K. (2021) 'Cloud resource scheduling with deep reinforcement learning and imitation learning', *IEEE Internet of Things Journal*, Vol. 8, No. 5, pp.3576–3586.

Jia, Y-H., Chen, W-N., Yuan, H., Gu, T., Zhang, H., Gao, Y. and Zhang, J. (2021) 'An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 51, No. 1, pp.634–649.

Konda, S., Panwar, L., Panigrahi, B., Kumar, R. and Gupta, V. (2021) 'Binary fireworks algorithm application for optimal schedule of electric vehicle reserve in traditional and restructured electricity markets', *International Journal of Bio-inspired Computation*, Vol. 18, No. 1, pp.38–48.

Laili, Y., Lin, S. and Tang, D. (2020) 'Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment', *Robotics and Computer-Integrated Manufacturing*, Vol. 61, p.101850.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K. and Ostrovski, G. (2015) 'Human-level control through deep reinforcement learning', *Nature*, Vol. 518, No. 7540, pp.529–533.

Muteeh, A.S.M.T.M. (2021) 'MRLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization', *Cluster Computing*, Vol. 24, pp.3135–3145.

Sahni, J. and Vidyarthi, D.P. (2018) 'A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment', *IEEE Transactions on Cloud Computing*, Vol. 6, No. 1, pp.2–18.

Srinivas, N. and Deb, K. (1994) 'Multiobjective function optimization using nondominated sorting genetic algorithms', *Evolutionary Computation*, Vol. 2, No. 3, pp.1301–1308.

Sutton, R.S.B.A.G. (1998) *Reinforcement Learning: An Introduction*, MIT Press, Boston.

Tong, Z., Chen, H., Deng, X., Li, K. and Li, K. (2020) 'A scheduling scheme in the cloud computing environment using deep q-learning', *Information Sciences*, Vol. 512, pp.1170–1191.

Topcuoglu, H., Hariri, S. and Wu, M-Y. (2002) 'Performance-effective and low-complexity task scheduling for heterogeneous computing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 2, pp.260–274.

Verma, A. and Kaushal, S. (2015) 'Cost-time efficient scheduling plan for executing workflows in the cloud', *Journal of Grid Computing*, Vol. 13, No. 4, pp.1–12.

Versluis, L. and Iosup, A. (2021) 'A survey of domains in workflow scheduling in computing infrastructures: community and keyword analysis, emerging trends, and taxonomies', *Future Generation Computer Systems*, Vol. 123, pp.156–177.

Wang, S., Ding, Z. and Jiang, C. (2021) 'Elastic scheduling for microservice applications in clouds', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 1, pp.98–115.

Watkins, C. and Dayan, P. (1992) 'Q-learning', *Machine Learning*, Vol. 8, pp.279–292.

Zhang, Z., Cao, Y., Cui, Z., Zhang, W. and Chen, J. (2021) 'A many-objective optimization based intelligent intrusion detection algorithm for enhancing security of vehicular networks in 6G', *IEEE Transactions on Vehicular Technology*, Vol. 70, No. 6, pp.5234–5243.

Zhang, Z., Zhao, M., Wang, H., Cui, Z. and Zhang, W. (2022) 'An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty', *Information Science*, Vol. 583, pp.56–72.