Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Data-intensive workflow scheduling strategy based on deep reinforcement learning in multi-clouds

Shuo Zhang[1,2], Zhuofeng Zhao[1,2*], Chen Liu[1,2] and Shenghui Qin[1,2]

Abstract

With the increase development of Internet of Things devices, the data-intensive workflow has emerged as a new kinds of representation for IoT applications. Because most IoT systems are structured in multi-clouds environment and the data-intensive workflow has the characteristics of scattered data sources and distributed execution requirements at the cloud center and edge clouds, it brings many challenges to the scheduling of such workflow, such as data flow control management, data transmission scheduling, etc. Aiming at the execution constraints of business and technology and data transmission optimization of data-intensive workflow, a data-intensive workflow scheduling method based on deep reinforcement learning in multi-clouds is proposed. First, the execution constraints, edge node load and data transmission volume of IoT data workflow are modeled; then the data-intensive workflow is segmented with the consideration of business constraints and the first optimization goal of data transmission; besides, taking the workflow execution time and average load balancing as the secondary optimization goal, the improved DQN algorithm is used to schedule the workflow. Based on the DQN algorithm, the model reward function and action selection are redesigned and improved. The simulation results based on WorkflowSim show that, compared with MOPSO, NSGA-II, GTBGA and DQN, the algorithm proposed in this paper can effectively reduce the execution time of IoT data workflow under the condition of ensuring the execution constraints and load balancing of multi-clouds.

Keywords Data-intensive workflow, Deep-Q-network, Multi-objective optimization, Intensive learning

Introduction

With the rapid development of Internet of Things (IoT) technology, the data that collected by various IoT devices are constantly generated and accumulated. Then the data-intensive applications are produced. However, the IoT devices have difficulty processing data-intensive applications locally due to the limited resources. It is difficult to provide all resources and schedule services for IoT users only by relying on a single cloud.

The IoT architecture composed of the center cloud and the edge clouds can be seen as multi-clouds [1]. For the data generated by the edge devices of the Internet of Things, there are often corresponding storage devices in the edge cloud to meet the rapid response required by edge computing. However, due to the limited computing resources of the edge cloud, the center cloud is also needed as an important part to meet the processing requirements of massive data. The multi-clouds are composed of the central cloud and the edge clouds, which complete multiple data-intensive workflow scheduling tasks. Different from the traditional workflow, the data-intensive workflow in the Internet

*Correspondence:
Zhuofeng Zhao
edzhao@ncut.edu.cn
[1] School of Information, North China University of Technology, Beijing 100144, China
[2] Beijing Key Laboratory of Large-Scale Stream Data Integration and Analysis Technology, Beijing 100144, China

of Things environment has the characteristics of scattered data sources, large data scale and distributed execution at the multi-clouds.

When executing this kind of workflow in multi-clouds environment, many factors such as business constraints about data privacy and long-distance data transmission should be considered. There are many challenges in the data flow control management and the data transmission scheduling.

The existing methods to solve data-intensive workflow scheduling mainly include cloud computing scheduling strategy based on heuristic thought, cloud computing scheduling strategy based on segmentation thought and cloud computing scheduling strategy based on reinforcement learning [2]. The heterogeneous distributed resource environment of cloud computing and the parallel task structure of data-intensive workflow together form a large state space. The reinforcement learning has powerful decision-making ability when dealing with the complex space problem. Therefore, reinforcement learning is often used as a powerful means to solve scheduling problems in recent years [3].

However, applying reinforcement learning to solve the scheduling problem of data-intensive scientific workflow in cloud computing environment has the following difficulties:

On the one hand, the time and the cost of data-intensive workflow mainly come from the link transmission process. The way to reduce the link loss is to reduce the data dependence among data centers, which usually requires the segmentation of workflow structure. On the other hand, the state set of workflow scheduling is complex, and there is a problem of over-dimensionality. It is difficult to store all reward values in the form of a table. It is solved by generalization of state vectors by neural network, and the state values are extracted by deep reinforcement learning (DRL) technology, so as to achieve the goal of dimensionality reduction.

Therefore, this paper proposes a data-intensive workflow scheduling method based on deep reinforcement learning in multi-clouds. The contributions of this paper are summarized as follows:

(1) In order to deal with the scheduling problem of data-intensive workflow, this paper proposes a method of workflow segmentation to reduce the data transmission between partitions which will be deployed in cloud center and edge cloud. By dividing the original workflow into several blocks with similar scale and low data dependence, the algorithm provides a certain environmental state model foundation for the deep reinforcement learning scheduling algorithm in the subsequent chapters.
(2) In this paper, deep neural network is introduced into reinforcement learning, and it is used to train reinforcement learning process. Based on the DQN algorithm, the idea of bias correction is introduced to calculate the variance of the current state reward to solve the problem of overestimation of Q value. In addition, the reward function is improved so that the workflow scheduling results converge to a stable correlated equilibrium policy.
(3) Finally, the open source WorkflowSim simulation environment is used to evaluate the proposed method. Compared with the traditional workflow scheduling method, the experimental results show that the proposed method can effectively improve the workflow execution time and load balancing.

The rest of this paper is organized as follows. The second section introduces the related work of this paper. In the third section, the related definitions involved in scheduling method and the segmentation method of data-intensive workflow are given. In the fourth section, the detailed method of workflow scheduling strategy is designed. Then, in the fifth section, the effectiveness of this method is verified by experiments based on WordflowSim simulation environment. Finally, the sixth section summarizes the full text.

Related work

The algorithm based on heuristic idea can efficiently find the approximate optimal solution of the workflow scheduling problem in cloud computing, and it is the mainstream type of cloud computing scheduling method in recent years. The basic prototypes are ant colony optimization (ACO), particle swarm optimization (PSO), genetic algorithm (GA), etc. [4]. Literature [5] puts forward the earliest completion time algorithm (HEFT), which is a constructive heuristic algorithm. The algorithm first sets the priority of each task in the workflow based on the average execution cost and the average transmission cost. Then allocates resources for the tasks according to the task priority and the earliest completion time of the task on the virtual machine. Literature [6] puts forward the big cuckoo algorithm, which imitates cuckoo's sojourning behavior. It aims at minimizing the turnaround time and maximizing the resource utilization rate. However, this algorithm fails to take into account the interaction of big data and is not suitable for data-intensive workflow. Literature [7] puts forward a multi-objective artificial bee colony algorithm, which is a swarm intelligence algorithm that can reduce energy consumption, execution time and cost respectively and improve resource utilization, but it does not discuss mutually exclusive performance indicators, such

as execution time and overall cost, and provides a compromise solution. Literature [8] puts forward a hybrid particle swarm optimization (PSO) HEFT algorithm, which focuses on solving the problem of high energy consumption in the process of workflow scheduling in cloud computing system, and it can obtain a scheduling solution that balances the scheduling quality and energy consumption, but this algorithm is not suitable for dealing with scientific workflow scheduling problems oriented to data flow. This kind of algorithm can find the feasible solution under the constraint conditions, but because it can't predict the deviation between the feasible solution and the optimal solution, the convergence speed is slow, and it often falls into the local optimal solution in the process of solving, so it is difficult to meet the task requirements of low latency.

When the workflow brings a heavy burden to the data link, researchers usually use graph segmentation to minimize the data traffic between the blocks, so as to reduce the data coupling within the secondary workflow, thus reducing the link load between data centers [9]. There are two important principles for segmentation of workflow flow graph on cloud computing platform, one is to make the data dependency between segmented subgraphs as small as possible, which can give full play to the advantages of distributed parallel computing of cloud computing; The second is to make the scale of each block as balanced as possible, which can avoid the short-board effect of workflow and improve the system performance. Literature [10–12] offload computing-intensive tasks to the edge server or cloud for processing, which maximizes the quality of user experience under resource constraints. Literature [13] use cuckoo search (CS) to segment the workflow, and finally the decision tree is used to allocate resources. Although this method can accelerate the iterative convergence and shorten the execution time, the selected fitness function can't describe the segmentation result well.

In recent years, reinforcement learning is often used as a powerful means to solve scheduling problems. By using the excellent decision-making ability of reinforcement learning to solve scheduling problems in complex edge environments, the convergence speed can be accelerated by constantly correcting the deviation of feasible solutions and better solutions. Literature [14] uses Q-learning algorithm to match resources in online dynamic scheduling. This method is oriented to unrelated tasks and it can obviously shorten the average response time of tasks. However, it is not suitable for workflow problems with priority or dependence, and it is difficult to predict and classify the upcoming tasks. Literature [15] uses reinforcement learning method to optimize the scheduling of memory controller, which improves the running state of application and bandwidth utilization. Finally, cerebellar neural network is used to reduce the dimension of state space, but it is not suitable for data-intensive workflow cloud scheduling. In literature [16], in order to improve the task processing efficiency for Internet of Vehicles (IoV),the paper design a CORA algorithm and use the Markov decision process model for formulating the dynamic optimization problem. In literature [17], the author developed a scheduling algorithm based on pointer network and reinforcement learning method. In the state set of the algorithm, parameters such as execution time, virtual machine failure probability, communication cost and associated tasks were defined, and a state neural network based on these parameters was analyzed and constructed. Literature [18] uses the game-based method to offload the computing-intensive tasks to achieve the goal of minimizing user costs and maximizing server profits.

Due to the limitations of reinforcement learning itself, it can't deal with the problem of high maintenance and continuity [19]. Deep learning method focuses on the expression of perception and input, and it is good at discovering the characteristics of data. Because deep learning can make up for the shortcomings of reinforcement learning, deep reinforcement learning (DRL) uses deep neural network's ability to capture environmental characteristics and the decision-making ability of RL can solve complex system control problems, and it can use edge nodes as intelligent agents to learn scheduling strategies without global information about the environment. Aiming at the data transmission overhead of data-intensive workflow, as well as the optimization objectives of workflow scheduling, such as execution time and load balance, etc., this paper studies the workflow scheduling algorithm based on deep reinforcement learning.

Data-intensive workflow segmentation method for Multi-Clouds execute scheduling and related definitions

When dealing with data-intensive workflow, because the data link needs to transmit a large amount of data, the overall consumption mostly comes from the cost of data transmission [20]. In addition, the deployment positions of cloud-edge nodes are scattered in the edge cloud, and the dependencies among tasks of workflow are complex. In order to deal with the scheduling problem of data-intensive workflow, this paper proposes a method of workflow segmentation to reduce the data transmission between partitions. Through the partition scheduling method based on constraints, the goal of minimizing the execution time of data-intensive workflow is achieved.

Related definitions

Data transmission between tasks

In this paper, workflow is represented based on directed acyclic graph (DAG), in which nodes are composed of tasks and edges represent their dependencies. The workflow before segmentation is recorded as ODAG, as shown in Fig. 1:

ODAG=(T,E), where t is the set of task vertices, and $T = \{T_1, T_2, ..., T_n\}$ indicates that cloud workflow consists of n dependent tasks; E is a set of directed edges.

$E = \left\{ \left(T_i, T_j, Data_{ij}\right) : \left(T_i, T_j\right) \in T \right\}$,$(T_i,T_j,Data_{ij})$ indicates the dependency between tasks. T_i is the predecessor of T_j, T_j is the successor of T_i, and $Data_{ij}$ indicates the amount of data transferred from T_i to T_j.

Data transmission between sub-workflows

After the data-intensive workflow is divided, sub-workflows are formed, which are also described by directed acyclic graph, which is called SDAG. Each sub-workflow is regarded as a vertex, and the dependency among sub-workflows is regarded as an edge.

Thus simplifying ODAG, as shown in Fig. 2. Write it as OG = (P,R), where P is the set of sub-workflows, and $P = \{P_1, P_2, ..., P_m\}$ indicates that the original work consists of m sub-workflows; R is the set of edges representing the dependency among sub-workflows. $R = \{(P_i, P_j, Data_{ij}) : (P_i, P_j) \in P\}$. $(P_i, P_j, Data_{ij})$ represents the dependency relationship among sub-workflows, and P_i is the predecessor of P_j, which is the successor of P_j. $Data_{ij}$ represents the amount of data that the P_i passes to the the P_j.

Execution time

The data-intensive execution time before segmentation depends on the amount of data transmission and the computing power of node resources between its sub-workflows Each sub-process will be assigned to a virtual machine in a cloud center or edge node, and each sub-process has a start time and an end time, which are marked as $sched(P_i) = (vm_j, SP_{P_i}, EP_{P_i})$, vm_j indicate that sub-workflow P_i is executed on virtual machine of vm_j. SP_{P_i} is the start time of P_i, EP_{P_i} is the end time of P_i. SP_{P_i} is determined by the time when all predecessor sub-workflows of P_i are executed and data is transmitted to the virtual machine where sub-workflow P_j is located. EP_{P_i} is the execution time of P_i and plus its start time SP_{P_i}, the formula is as follows:

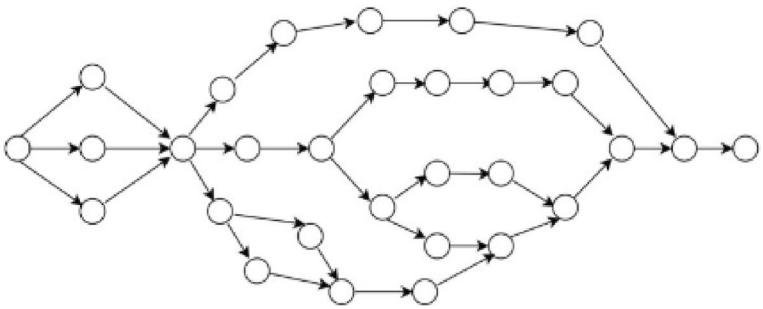

Fig. 1 Original data workflow DAG

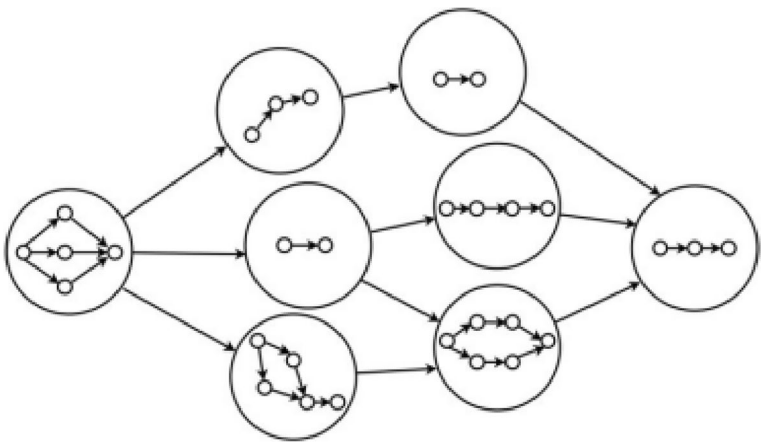

Fig. 2 Segmented data workflow ODAG

$$\mathrm{EP}_{p_i} = SP_{P_i} + \frac{work(\mathrm{P_i})}{\mathrm{P_{vm_j}}}$$

$$SP_{P_i} = \max\{EP_{P_h} + \frac{Data(P_{hi})}{b} : P_h \in pre(P_i)\}$$

EP_{P_h} is the end time of the precursor workflow $\mathrm{P_h}$, $\frac{Data(P_{hi})}{b}$ is the time when data is transmitted from $\mathrm{P_h}$ to P_i. b is the data transmission rate between nodes. $work(P_i)$ is the instruction number of workflow P_i. P_{vm_j} is the number of instructions processed per second for vm_j. The time period from the first sub-workflow to the last sub-workflow is called the completion time of the whole IoT data workflow, so the completion time of the whole IoT data workflow is as follows:

$$makespan = \max\left\{EP_{P_h} : P_i \in P\right\}$$

Node load balance degree

Cloud and edge nodes, as the execution carriers of data-intensive workflow, must ensure their stable and normal operation, and node load balancing can effectively guarantee the normal and stable operation of the whole system [21]. Therefore, it is of great significance to ensure the load balancing of each node in the scheduling process for the stability of workflow execution. The load balancing is shown in Fig. 3. n_i represents the computing node, and P_i represents the sub-workflow.

Compute resource node j (j = 1,2,…,K; K is the total number of resource nodes) to represent the frequency of the resource nodes executing sub-workflows, which is denoted by N_j. Assuming that each virtual machine in each compute node has the same computing power, the calculation formula of resource utilization of a single compute node is:

$$\mathrm{r} = \frac{\sum_{j=1}^{k} N_j}{K \bullet \max\limits_{j \le k} N_j} \times 100\%$$

Here, it is expressed by the degree of equilibrium. In the process of optimization, the value of r needs to be as close to 1 as possible. In addition, in this paper, the average load of multiple computing nodes (cloud center and edge nodes), $\frac{r}{N}$ is used as the optimization objective (n is the number of nodes).

Data-intensive workflow segmentation algorithm

Business constraints

Business constraint refers to the data privacy protection constraint, that is, the data source or running scenario of the constrained task is fixed and must be executed on one or more specified nodes [22]. In this paper, based on business bundle, data-intensive workflow segmentation optimization and scheduling optimization are carried out. The locations in the scheduling range are one-hot encoding, and each geographical location is a string of length N, where N is the number of all locations, and only one number in the string is 1, and all other numbers are 0, as shown in Fig. 4, n_i represents the computing node, P_i represents the sub-workflow, and D_i is the corresponding dependent data. In this representation, if the position of the required data determines the position of the computing task, the following formula must be satisfied:

$$location_{data} \bullet location_{task} = 1$$

Where $\mathrm{location}_{data}$ is the position vector of the constrained data set and $\mathrm{location}_{task}$ is the position vector corresponding to the data processing task.

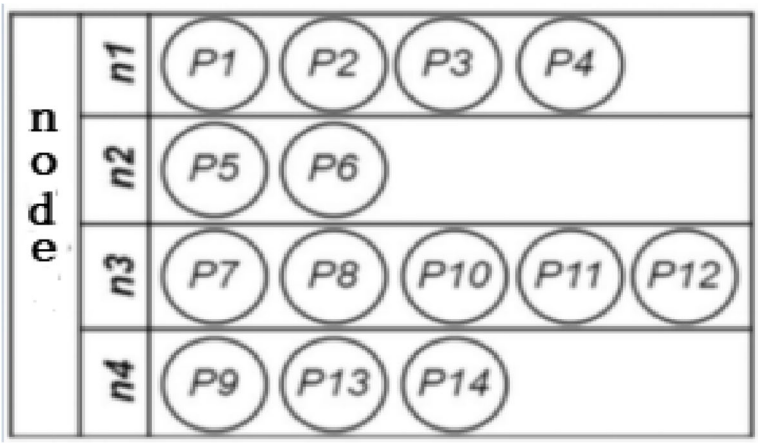

Fig. 3 Node load balancing diagram

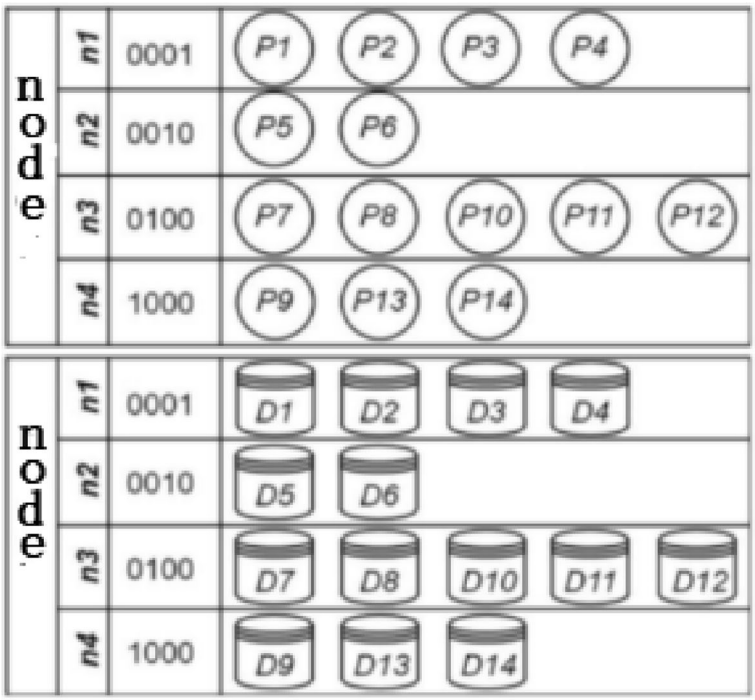

Fig. 4 Business constraint rules

Segmentation algorithm

Business constraints are the premise of segmentation location optimization. When the tasks of location$_{task}$ are equal, these tasks belong to the same set of constraints. On the basis of constraints, the following segmentation optimization algorithm is given, and the business constraint set is further optimized according to the data transmission volume to obtain a new task set.

The algorithm steps are as follows:

(1) Incorporate unconstrained tasks related to constrained tasks into constrained task sets: take the average value of internal data transmission in each constrained task set as a threshold value w of data transmission, and traverse unconstrained tasks in ODAG from tasks in constrained task sets. If the data transmission between two adjacent tasks is greater than the threshold value w, merge subsequent tasks into the constrained task set, and continue to traverse. If it is less than w, the traversal is finished. Traverse from the next constraint task set and repeat the above steps.
(2) The unconnected related unconstrained tasks form a new task set: traverse from the start node of ODAG, and form a new task set with continuous unconstrained tasks.
(3) Further optimizing the division position in the task set: if the task set is allowed to be scheduled to multiple service nodes, the set can be further divided into multiple sets according to the threshold W.

Data-intensive workflow scheduling strategy

This paper presents a workflow scheduling method based on DQN algorithm. On the basis of DQN algorithm, the model reward function is redesigned and improved according to the characteristics of the research problem.

Multi-objective optimal scheduling

In order to promote the development of multi-objective optimal scheduling method based on deep reinforcement learning, this paper puts forward the following assumptions [23]:

① Each task can only be performed by one cloud host;
② The running time of the task is the time interval between the start and end of the task;
③ The delay time of resource supply or cancellation is not considered;
④ Do not consider the delay time of transmission between tasks.

In this paper, two QoS indexes, namely the makespan of workflow and load balance, are considered. As the goal of cloud workflow scheduling, it is a bi-objective

optimization problem. The goal of scheduling optimization algorithm can be expressed as follows:

$$\begin{cases} f(x) = \min(Tw_{total}, Sl_{total}) \\ S(p_i) = r_m\, p_i \in P, r_m \in dc_p \end{cases}$$

Among them, Tw_{total} is the execution time of IoT data workflow and Sl_{total} is the average load of the system; $S(p_i)$ is the constraint condition of the algorithm, and p_i is the sub-workflow with business constraints and dc_p is the service node assigned,r_m is the resources required by sub-workflow p_i on service node dc_p.

These two optimization objectives are abstracted into two agents respectively, each agent is an agent based on DQN algorithm, and carries out adaptive learning and self-optimization process through interaction with the environment and other agents.

Workflow scheduling method based on improved DQN

DQN

DQN algorithm is a popular method in the field of deep reinforcement learning. Its main modules include: environment module, loss function, experience replay module and two neural networks with the same structure but different parameters, namely estimated value network and target value network [24].

The DQN algorithm learns the action value function Q* corresponding to the optimal strategy by minimizing the loss,

$$l(\theta) = IE_{s,a,r,s'}\left[\left(Q^*(s,a|\theta) - y\right)^2\right]$$
$$y = r + \delta \max_{a'}\left(Q^*\left(s', a'|\theta^-\right)\right)$$

Among them, y is the objective Q function, and its parameters are updated periodically with the latest ones θ, which is helpful for stable learning.

DQN uses Q-table to store the Q values of each state-action pair, and DQN uses neural network to extract complex features and analyze them to generate Q values [25]. The estimated Q value network is used to predict the estimated Q value. Its input comes from the latest parameters of the current environment, and the parameters will be updated every iteration. θ is weight, $Q^*(s,a|\theta)$ is used to represent the output of the current estimated value network. The input parameters of the target Q value network are updated every once in a while. $\max_{a'}(Q^*(s\prime, a\prime|\theta^-))$ indicates the output of the target value network. The training goal of the neural network is to optimize the loss function constructed by these two Q values, and then update the parameters of the estimated Q value network by using the method of random gradient descent through back propagation. Every certain number of iterations, the parameters of the estimated Q value network will be copied to the target Q value network regularly. To some extent, it reduces the correlation between the estimated Q value and the target Q value, making divergence or oscillation more impossible, thus improving the stability of the algorithm [26, 27].

The use of neural network module in DQN overcomes the high-dimensional data disaster of single-agent reinforcement learning. and balances the contradiction between exploration and utilization to some extent through the use of target value network, experience playback pool and exploration mechanism based on εO method.

DQN-RL

In this paper, we propose a DQN method based on bias correction. Q-values are obtained from multiple historical online value network models and online value network outputs. The variance of these multiple current state rewards is calculated, and then the bias correction term is calculated based on the variance and applied to the target Q-value formula, which solves the problem of Q-value overestimation to some extent.

The formula for calculating the target Q value in DQN is shown in the following equation:

$$y_t = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-)$$

The improved Q-value calculation formula:

$$y_{DQN-DL} = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-) - B(s,a,r)$$

In the above equation, r_t shows the estimate of the tth immediate reward. θ shows the parameters of the saved historical online value network model. a shows the saved actions in the empirical data. $B(s,a,r)$ is the modified bias correction term.

In multi-agent learning systems, it is usually faced with the challenges of difficult determination of learning objectives, unstable learning problems and coordination of processing. In this paper, the reward function is improved so that the workflow scheduling results converge to a stable correlated equilibrium policy.

(1) State Space

The state space set in this paper is represented by a vector $Vector = [s_1, s_2, ..., s_i, ..., s_n]$ where n is the number of tasks in the workflow, the index in the vector represents the $ID_{t_{id}}$ of each task, s_i is is an integer represents the state of the ith task, and -1 represents that the task has been executed. -2 means that the task can be executed; -3 means that the predecessor node of the task has not been executed, that is, the execution conditions

are not met;0 ~ m represents being executed by the virtual machine,and the value is the id of virtual machine [28].

(2) Reward function

A suitable reward function design can ensure the stability and convergence of the algorithm in multi-agent learning scenarios. For the agent with maximum completion time, the reward function designed in this paper is as follows,

$$w_1 = [\frac{ET_{k,i,j}(a) - (makespan\prime - makespan)}{ET_{k,i,j}(a)}]^3$$

The reward function of load balancing is as follows,

$$w_2 = [\frac{ET_{k,i,j}(a) - (r\prime - r)}{ET_{k,i,j}(a)}]$$

The value ranges of the reward function of execution time w_1 and the reward function of load balancing w_2 both fall within [0,1], which means that the execution time is updated to make the value of the increased execution time as small as possible, and the corresponding reward value is closer to 1; Otherwise, it approaches 0. Similarly, the second formula represents that the smaller the added value of load balancing is, the more desirable its strategy is, and the closer its reward value is to 1; Otherwise, there is no reward and the value is 0.

(3) Action selection

In the process of reinforcement learning, the action with the largest Q value will be selected every time, which is to use greedy strategy to perform action selection [29, 30]. However, in the initial stage of reinforcement learning, agents can't master the Q value, so they need to explore and choose unknown actions in a random way. After a period of learning, they can get a certain amount of Q value. However, at this time, whether to continue to explore unknown actions or make use of the current action with the largest Q value is the balance problem of exploration and utilization faced by reinforcement learning. In order to solve this problem, this paper uses the variable εO strategy, that is, at the beginning, s is set to be larger, such as 0.9, to give the model more opportunities to explore; With the increase of training rounds, the learning ability of the model becomes stronger, and the updated state action value becomes better. The value of s is gradually reduced, and the learned Q value is more used to choose the best behavior.

Simulation analysis

Experimental data set

In this paper, CyberShake is used as the workflow for experiments. CyberShake workflow is usually used to process seismic data, and was originally used by Southern California Earthquake Center [31]. As shown in Fig. 5. Cybershake is a computation-intensive and data-intensive workflow. The number of task nodes can be different, and it can also handle large data sets, which is very suitable for verifying the effectiveness of the algorithm proposed in this paper. And each color represents a different workflow.

Experimental environment and parameter setting

Experimental environment

Through the effective integration of workflow simulation platform WorkflowSim and cloud computing environment simulation platform CloudSim, the scheduling simulation of workflow in edge cloud is carried out, and the software development environment is JDK1.7.0; Hardware development environment is inter (r) core (TM) i7-5600 CPU @ 2.60ghz, and memory is 16 GB.

Data center and virtual machine configuration in WorkflowSim

The experiment was conducted in five data centers in different geographical locations, with the first four data centers representing four edge computing nodes and the remaining one representing the cloud computing center. Each data center in the first four data centers is equipped with one host, and the fifth data center is equipped with three hosts, as shown in Table 1. The parameter settings of edge nodes and cloud computing centers are shown in Table 2. At the same time, three types of virtual machines, 10 in each type, were randomly allocated to five data centers (Table 3).

Parameter setting

Training parameter settings are shown in Table 4.

Comparison method

In this paper, MOPSO [32], NSGA-II [33] GTBGA [34] and DQN [35] are used as benchmark comparison algorithms.

(1) The GTBGA combines game theory and greedy strategy. Firstly, the tasks of different scientific workflows are divided into task packages that can be executed at different stages. The game between phased assignable tasks and available cloud hosts is balanced and matched. Because there are multiple

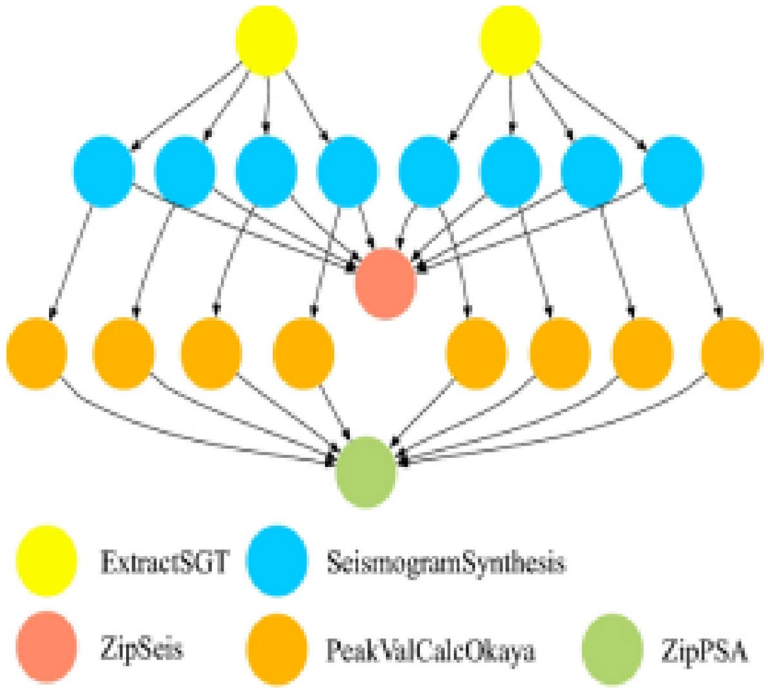

Fig. 5 CyberShake

Table 1 Configuration of five data centers

Data Center	Number of hosts	Broadband	Remark
Datacenter_0	1	1.5e7	Edge
Datacenter_1	1	1.5e7	Edge
Datacenter_2	1	1.5e7	Edge
Datacenter_3	1	1.5e7	Edge
Datacenter_4	3	0.75e7	Cloud

Table 2 Parameter configuration of edge nodes and cloud computing centers

Parameter	Edge node	Cloud node
Mips	3000	3000
Ram	1800	2048
Storage	10,000	10,000
BW	10,000	10,000

Table 3 Parameter configuration of virtual machines

Parameter	1	2	3
Size	1100	800	1100
Mips	512	512	512
Ram	1000	800	1200
Bw	1000	1200	800

Table 4 Parameter configuration of training

Parameters	Value
Learning rate	0.002
Reward attenuation rate	0.9
Greed factor	0.7
Max	0.95
Growth rate	1e-5
Experience pool storage size	10,000
Minibatch	128
replace_target_iter	500
Activation function	Relu
Hidden layer	7

stages, and the scheduling process only considers the optimization of a single stage, it is greedy.

(2) The MOPSO is a multi-objective optimization algorithm based on heuristics, combined with Pareto optimization technology. On the basis of PSO algorithm, MOPSO added an external Archive and combined with special mutation operations to find the Pareto optimal solution set.

(3) NSGA-II is a multi-objective optimization algorithm based on heuristics. NSGA-II is an improved algorithm based on NSGA, which ensures the diversity of solutions by using elite selection strategy and crowding distance and does not have to rely on shared parameters.
(4) The DQN method combines Q Learning with deep learning, uses the deep network to approximate the action value function. And the DQN uses the experience playback mechanism and the target network to stabilize the training process.

Experimental results

To evaluate the effectiveness of the scheduling algorithm proposed in this paper, it is compared with MOPSO, NSGA-II, GTBGA and DQN. Then, the execution time of workflow and load balance of nodes are taken as

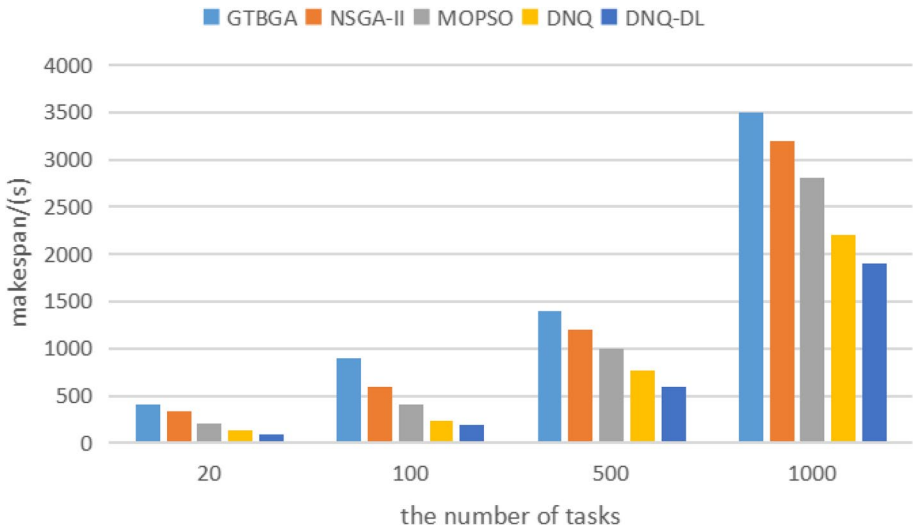

Fig. 6 Comparison of execution time

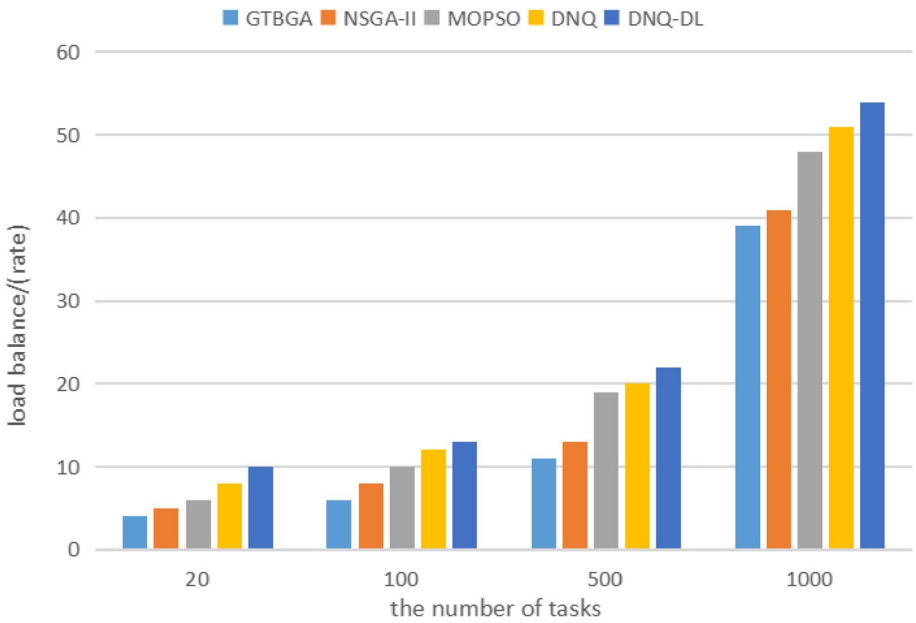

Fig. 7 Comparison of load balancing

optimization objectives. Comparative experimental results are shown in Figs. 6 and 7.

The comparison between this method and the baseline method in workflow execution time is shown in Fig. 6. The abscissa is the number of tasks. The ordinate is the execution time,and the unit is seconds.

As can be seen from Fig. 6, compared with DQN algorithm, the makespan of workflow tasks deployed by DQN-DL algorithm saves 21% on average, 40% on average compared with MOPSO algorithm, 50% on average compared with NSGA-II algorithm and 57% on average compared with GTBGA algorithm. It can be seen from the graph that the method proposed in this paper has the shortest execution time, with the shortest execution time of 100, and the best effect.

The comparison between this method and the baseline method in load balancing is shown in Fig. 7. The abscissa is the number of tasks, the ordinate is the degree of load balance, and the unit is rate. As can be seen from Fig. 7, compared with DQN algorithm, the load balance of workflow tasks deployed by DQN-DL algorithm has increased by 5% on average, by 11% compared with MOPSO algorithm, by 24% compared with NSGA-II algorithm and by 27% compared with GTBGA algorithm. To sum up, in the aspect of server load balancing of workflow tasks, the DQN-DL algorithm is better than the other four algorithms, thus better ensuring the stability of computing nodes.

In a word, the scheduling scheme given by the DQN-RL algorithm proposed in this paper is far superior to other comparison algorithms in terms of the execution time of workflow and the load balance of nodes. All of them have such good performance, precisely because of the excellent performance of deep reinforcement learning in this large-scale decision-making problem, and its decision-making ability and the ability to find the optimal solution are far superior to the traditional heuristic algorithm.

Conclusion

In the cloud-edge collaborative environment, IoT data workflow has a large amount of data and scattered data sources, so the data dependence among tasks of IoT data workflow is complex, and data transmission is inevitable during scheduling. This paper adopts the method based on deep reinforcement learning to optimize the multi-objective scheduling of data-intensive workflows, first divides the data-intensive workflows, and then uses the improved DQN algorithm to schedule multiple workflows. Through the experimental evaluation, this method can effectively optimize the execution time of data workflow adjustment, effectively improve the service quality, and make the average load of each node more balanced, making the system work more stable.

Abbreviations

DQN	Deep-Q-network
ACO	Ant colony optimization
PSO	Particle swarm optimization
GA	Genetic algorithm

Acknowledgements

The authors would like to thank all the staf and students of school of North China University of Technology for contribution during this research process.

Authors' contributions

Shuo Zhang: Thesis Writing and Algorithm Experiment; Zhuofeng Zhao: topic determination and introduction; Chen Liu: Check the paper; Shenghui Qin: Write the third chapter.

Funding

The work of this paper are supported by the Key-Area Research and Development Program of Guangzhou City (202206030009) and the Beijing Municipal Natural Science Foundation (No. 4202021).

Availability of data and materials

The data used during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Consent has been granted by all authors and there is no confict.

Competing interests

The authors declare no competing interests.

Received: 31 December 2022 Accepted: 14 August 2023
Published online: 26 August 2023

References

1. Huang J, Gao H, Wan S et al (2023) AoI-aware energy control and computation offloading for industrial IoT. Futur Gener Comput Syst 139:29–37
2. Huang J, Zhang C, Zhang J (2020) A multi-queue approach of energy efficient task scheduling for sensor hubs. Chin J Electron 29(2):242–247
3. Shyalika C, Silva T, Karunananda A (2020) Reinforcement learning in dynamic task scheduling: a review. SN Comput Sci 1:1–17
4. Masdari M, ValiKardan S, Shahi Z et al (2016) Towards workflow scheduling in cloud computing: a comprehensive analysis. J Netw Comput Appl 66:64–82
5. Dubey K, Kumar M, Sharma SC (2018) Modified HEFT algorithm for task scheduling in cloud environment. Procedia Comput Sci 125:725–732
6. Navimipour NJ, Milani FS (2015) Task scheduling in the cloud computing based on the cuckoo search algorithm. Int J Model Optim 5:44–47
7. Gao KZ, Suganthan PN, Pan QK, Chua TJ, Chong CS, Cai TX (2016) An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. Expert Syst Appl 65:52–67
8. Wang F, Zhang H, Li K et al (2018) A hybrid particle swarm optimization algorithm using adaptive learning strategy. Inf Sci 436:162–177
9. Pei S, Zhang Q, Cheng X (2020) Workflow scheduling using graph segmentation and reinforcement learning. Int J Perform Eng 16(8)
10. Chen Y, Zhao J, Wu Y, Huang Y, Shen XS (2022) QoE-aware decentralized task Offloading and resource allocation for end-edge-cloud systems: a

game-theoretical approach. IEEE Trans Mob Comput. https://ieeexplore.ieee.org/document/9954914
11. CHEN Ying, HU Jintao, ZHAO Jie, et al (2023) QoS-Aware Computation offloading in LEO satellite edge computing for IoT: a game-theoretical approach. Chin J Electron. https://cje.ejournal.org.cn/article/doi/10.23919/cje.2022.00.412
12. Ying Chen, Jie Zhao, Xiaokang Zhou, Lianyong Qi, Xiaolong Xu, Jiwei Huang (2023) A distributed game theoretical approach for credibility-guaranteed multimedia data offloading in MEC. Inf Sci 644:0020–0255
13. Alawad NA, Abed-alguni BH (2021) Discrete island-based cuckoo search with highly disruptive polynomial mutation and opposition-based learning strategy for scheduling of workflow applications in cloud environments. Arab J Sci Eng 46(4):3213–3233
14. Kaur A, Singh P, Singh Batth R et al (2022) Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud. Softw Pract Exp 52(3):689–709
15. Chen Y, WG U, Xu J, Zhang Y, Min G (2023) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. Chin Commun 1–12. https://ieeexplore.ieee.org/abstract/document/10122834
16. Huang J, Wan J, Lv B, Ye Q, Chen Y (2023) Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via Deep reinforcement learning. IEEE Syst J 17(2):2500–2511
17. AL-Tam F, Mazayev A, Correia N, Rodriguez J (2020) Radio resource scheduling with deep pointer networks and reinforcement Learning. 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design ofCommunication Links and Networks (CAMAD). IEEE, Pisa, Italy, p. 1-6
18. Ying Chen, Jie Zhao, Jintao Hu, Shaohua Wan, Jiwei Huang (2023) Distributed task offloading and resource purchasing in NOMA-enabled mobile edge computing: hierarchical game theoretical approaches. ACM Trans Embed Comput Syst 1539–9087. https://dl.acm.org/doi/abs/10.1145/3597023
19. Ling N, Wang K, He Y, et al (2021) Rt-mdl: supporting real-time mixed deep learning tasks on edge platforms. In: Proceedings of the 19th ACM conference on embedded networked sensor systems. pp 1–14
20. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
21. Ghomi EJ, Rahmani AM, Qader NN (2017) Load-balancing algorithms in cloud computing: a survey. J Netw Comput Appl 88:50–71
22. Monakova G, Leymann F (2013) Workflow ART: a framework for multidimensional workflow analysis. Enterp Inf Syst 7(1):133–166
23. Quan Z, Wang Y, Ji Z (2022) Multi-objective optimization scheduling for manufacturing process based on virtual workflow models. Appl Soft Comput 122:108786
24. Wang Y, Liu H, Zheng W et al (2019) Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. IEEE Access 7:39974–39982
25. Liu H, Ma Y, Chen P, et al (2020) Scheduling multi-workflows over-edge computing resources with time-varying performance, A novel probability-mass function and DQN-based approach. In: Web Services–ICWS 2020: 27th International Conference,Springer, Cham, Honolulu, p. 197–209
26. Wang Y, Jiang J, Xia Y, et al (2018) A multi-stage dynamic game-theoretic approach for multi-workflow scheduling on heterogeneous virtual machines from multiple infrastructure-as-a-service clouds. In: International conference on services computing (SCC). Springer, Zhuhai, pp 137–152
27. Tong Z, Chen H, Deng X et al (2020) A scheduling scheme in the cloud computing environment using deep Q-learning. Inf Sci 512:1170–1191
28. Çatal O, Wauthier S, De Boom C et al (2020) Learning generative state space models for active inference. Front Comput Neurosci 14:574372
29. Huang L, Bi S, Zhang YJ (2020) Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Trans Mobile Comput 19(11):2581–2593
30. Meng F, Chen P, Wu L (2019) Power allocation in multi-user cellular networks with deep Q learning approach. In: Proc. IEEE Int. Conf. Commun. pp 1–6
31. Jain A, Kumari R (2017) A review on comparison of workflow scheduling algorithms with scientific workflows. In:Proceedings of International Conference on Communication and Networks. vol 508. Springer, Singapore, p. 613–622. https://link.springer.com/chapter/10.1007/978-981-10-2750-5_63
32. Rehman A, Hussain SS, urRehman Z et al (2019) Multi-objective approach of energy efficient workflow scheduling in cloud environments. Concurr Comput Pract Exp 31(8):e4949
33. Li H, Wang B, Yuan Y et al (2021) Scoring and dynamic hierarchy-based NSGA-II for multiobjective workflow scheduling in the cloud. IEEE Trans Autom Sci Eng 19(2):982–993
34. Dong T, Xue F, Xiao C, Zhang J (2021) Deep reinforcement learning for dynamic workflow scheduling in cloud environment. 2021 IEEE International Conference on Services Computing (SCC), Chicago, IL, USA. p. 107–115
35. Huo D, Wu H, Wang B, et al (2022) A DQN-based workflow task assignment approach in cloud-fog cooperative considering terminal mobility. In: The 6th International Conference on Control Engineering and Artificial Intelligence. pp 78–82

Publisher's Note