Received August 25, 2019, accepted September 9, 2019, date of publication September 23, 2019, date of current version October 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943193

A Novel Task Provisioning Approach Fusing Reinforcement Learning for Big Data

YONGYI CHENG AND GAOCHAO XU
College of Computer Science and Technology, Jilin University, Changchun 130012, China

Corresponding author: Gaochao Xu (xugc@jlu.edu.cn)

This work was supported in part by the Jilin Provincial Industrial Innovation Special Foundation Project under Grant 2017C028-4, and in part by the National Natural Science Foundation of China under Grant 61972054.

ABSTRACT The large-scale tasks processing for big data using cloud computing has become a hot research topic. Most of previous work on task processing is directly customized and achieved through existing methods. It may result in relatively more system response time, high algorithm complexity and resource waste, etc. Based on this argument, aiming at realizing overall load balancing, bandwidth cost minimization and energy conservation while satisfying resource requirements, a novel large-scale tasks processing approach called TOPE (Two-phase Optimization for Parallel Execution) is developed. The deep reinforcement learning model is designed for virtual link mapping decisions. We treat whole network as a multi-agent system and the whole process of selecting each node's next hop node is formalized via Markov decision process. We train the learning agent by deep neural network to store parameters of deep network model while approximating the value function, rather than tons of state-action values. The virtual node mapping is achieved by designed distributed multi-objective swarm intelligence to realize our two-phase optimization for task allocation in topology structure of Fat-tree. We provide experiments to show the ability of TOPE in analyzing task requests and infrastructure network. The superiority of TOPE for large-scale tasks processing is convincingly demonstrated by comparing with state-of-the-art approaches in cloud environment.

INDEX TERMS Large-scale tasks, big data, two-phase optimization, reinforcement learning, fat-tree.

I. INTRODUCTION

The problem of large-scale tasks processing for big data [1]–[3] has become a hot issue. Aiming at achieving the efficient task execution, the large-scale tasks processing based on cloud computing [4]–[8] has been utilized to relieve the workload of computing and data transmission for big data learning. An efficient task processing approach can achieve the reasonable task allocation and improve the resource utilization rate. On the contrary, the load imbalance of system may occur since there are limitations of some conventional approaches and the performance variation of resource in heterogeneous system environment, which results in the decrease of resource utilization rate and performance of data center. Thus it is necessary to explore an effective large-scale tasks processing approach in the course of big data learning.

Some separate domains have attracted much attention and developed rapidly. Nowadays, with the rapid growth and diversification of user requirements, the combination of various domains has become a trend. The motivation of this work is to achieve the combination of multiple domains in right of combining the distributed swarm intelligence [9] and reinforcement learning [10] to complete reasonable task allocation in large-scale tasks processing for big data learning. We aim to develop an effective task processing approach, which can achieve the efficient parallel execution of massive tasks with high resource utilization rate, low bandwidth cost and energy consumption.

Based on this motivation, we have proposed a novel large-scale tasks processing approach for big data learning. The architecture of fat-tree [11], [12] has been introduced to reduce the load of some links with less bandwidth resource and make the topology more stable. The full connection pattern significantly increases the throughput of system, which benefits to the raise of overall system performance. And this work takes advantage of the idea of "divide-and-conquer" to achieve the combination of distributed swarm intelligence and the deep reinforcement learning to allocate tasks onto

The associate editor coordinating the review of this manuscript and approving it for publication was Christos Verikoukis.

proper physical nodes in the substrate network, which conduces to the tons of tasks processing in big data learning. It has realized the overall load balancing and minimization of bandwidth cost as well as energy saving in cloud data center.

Our main contributions are as follows:

(1) Based on the situation of rapid growth and diversification of user requirements, a two-phase optimization methodology for large-scale task processing in big data environment is proposed;

(2) This work presents a virtual network mapping based large-scale tasks processing approach combining the deep reinforcement learning with distributed multi-objective swarm intelligence while reducing the additional computational overhead and energy consumption;

(3) The capability of TOPE in analyzing task requests and infrastructure network has been shown in the experiments. Comparing with state-of-the-art approaches, it convincingly demonstrates the superiority of TOPE for large-scale tasks processing in distributed environment.

The rest of this paper is organized as follows: the related work of the current methods achieving tasks processing for big data is introduced in Section II. And in Section III, the proposed problem is described. In Section IV, we formalize the proposed problem in this work and describe the optimization problem. In Section V, the design and implementation of the main algorithm are introduced in detail. Moreover, the origin of the idea of this work is expounded in Section VI. In Section VII, the experimental results are obtained by experiments, demonstrating that the proposed approach can effectively overcome the problem presented herein. Finally, we draw the conclusion of this paper and specify some future work in Section VIII.

II. RELATED WORK

In recent years, efforts are invested to mitigate the various problems in task processing for big data. In [13], Jin Xiaohong et al. have introduced the task allocation techniques with clustering and load balancing in the field of Internet to the field of image processing job allocation of alternative big data. Researchers design and realize a load balancing cluster architecture for the alternative big data. Moreover, an improved load balancing algorithm applicable to large-scale image processing has been proposed. In [14], Tsuguhito Hirai et al. have taken the efficiency of backup tasks into consideration. The task-scheduling server is modeled as a single-server queue, in which the server consists of a number of workers. The task has been split into subtasks, and each subtask is served by its own worker and an alternative distinct worker in the case that a task enters the server. It is explicitly derived that the task processing time distributions for the two cases that the subtask processing time of a worker obey Weibull or Pareto distribution. Jian-Hua Gu *et al.* [15] have proposed a cross-domain workflow scheduling system named Arana. In right of moving program close to data and integrating popular big data processing platform, this system enables users to complete computing with cross-domain data without transferring. In [16], an intermediary framework by using multiple cloud environments has been presented to provide streaming big data computing service with lower cost per load. In this framework, a pricing strategy has been presented to maximize the revenue of the multiple cloud intermediaries. In the extensive simulations, it can be observed that the proposed pricing strategy can bring higher revenue than other pricing methods. In [17], researchers seek to propose a predictive approach to task scheduling with the aim of reducing the overhead incurred in case of processing big data based on the cloud. A method of using classification in machine learning is presented as a tool for scheduling tasks and assigning them to VMs in the cloud environment. A comparative study has been undertaken to explore which brand of classifiers perform optimally in the given scenario. And then a number of classification algorithms such as Naive Bayes, Random Forest and K-Nearest Neighbor are then used to predict the VM best suited to a task in the test dataset. In [18], a Parallel Random Forest (PRF) algorithm is proposed for big data processing on the Apache Spark platform. It achieves optimization based on a hybrid approach combining data parallel and task-parallel optimization. From the perspective of data-parallel optimization, a vertical data-partitioning method has been performed to reduce the data communication cost effectively, and a data-multiplexing method is performed to allow the training dataset to be reused and diminish the volume of data. From the perspective of task-parallel optimization, a dual parallel approach is carried out in the training process of RF, and a task Directed Acyclic Graph (DAG) is created according to the parallel training process of PRF and the dependence of the Resilient Distributed Datasets (RDD) objects. Hereupon different task schedulers are invoked for the tasks in the DAG. Experimental results indicate the superiority and notable advantages of the PRF algorithm over the relevant algorithms implemented by Spark MLlib and other studies in terms of the classification accuracy, performance, and scalability. With the expansion of the scale of the random forest model and the Spark cluster, the advantage of the PRF algorithm becomes more obvious. In [19], a model of computation partitioning for stateful data in the dynamic environment is proposed that will improve performance. First, a model of stateful data streaming is constructed, and the method of computation partitioning in a dynamic environment is studied. A definition of direction and calculation of the segmentation scheme is developed, including single frame data flow, task scheduling and executing efficiency. Second, a computation partitioning method for single frame data flow has been proposed. The data parameters of the application model, the computation partitioning scheme, and the task and work order data stream model have been determined. Finally, the research verifies the effectiveness of single frame data in the application of the data stream.

Recent advances in reinforcement learning and even deep reinforcement learning have improved the effectiveness of task processing for the big data. In [20], researchers investigate the cost minimization problem of big data analytics on geo-distributed data centers connected to renewable energy sources with unpredictable capacity. They propose a job scheduling algorithm by combining reinforcement learning with neural network. What's more, two techniques are developed to enhance the performance of the proposal. Specifically, Random Pool Sampling (RPS) is proposed to retrain the neural network via accumulated training data, and a novel Unidirectional Bridge Network (UBN) structure is designed for further enhancing the training speed by using the historical knowledge stored in the trained neural network. In [21], a new model for large-scale adaptive service composition is proposed. The model integrates the knowledge of reinforcement learning for the problem of adaptability in a highly-dynamic environment and game theory used to coordinate agents' behavior for a common task. In particular, a multi-agent Q-learning algorithm for service composition based on this model is also proposed herein. In [22], Fanyu Bu has proposed a reinforcement learning-based intelligent scheduling algorithm for big data analysis by increasing the utilization and reducing the energy consumption of the processors. A reinforcement learning model has been designed to select an appropriate dynamic voltage and frequency scaling technique for configuring the voltage and frequency according to the current system state, which can improve the utilization and optimize the energy consumption effectively. Furthermore, a learning algorithm has been implemented to train the parameters of the reinforcement learning model. The proposed scheduling approach is able to improve the resource utilization and save the energy for big data analysis in communication systems when performing tasks on mobile computing devices with embedded systems. In [23], Ying He et al. have considered realistic time-varying channels as well as the channel is formulated as a finite-state Markov channel (FSMC). They propose a novel big data reinforcement learning approach since the complexity of the system is very high. Deep reinforcement learning is used herein to obtain the optimal interference alignment user selection policy in cache-enabled opportunistic interference alignment wireless networks. In this work, the deep reinforcement learning model based on deep Q-network is trained to approximate the value function as well as make reasonable virtual link mapping decisions. The virtual node mapping can be determined by the designed distributed multi-objective swarm intelligence method to achieve the two-phase optimization for large-scale tasks allocation in cloud environment.

III. THE PROPOSED PROBLEM

As shown in Fig. 1, the fat-tree topology structure has been introduced to construct the system model of the proposed TOPE approach in the distributed environment. It can relieve the workload of interaction between master nodes and users, as well as ease the workload of direct interaction between master nodes and slave nodes. Specifically, parallel tasks are allocated to the master nodes in the core layer in order to accomplish the task allocation and management for the slave nodes. The sub-master nodes in the edge layer are divided into k independent zones. Based on the characteristic of fat-tree, each of sub-master nodes in the edge layer possesses $k/4$ link connected to the master node in the core layer. Meanwhile it also connects to other two nodes in this zone. The failure of certain sub-master node in the zone will not results in the offline of a large number of servers since the sub-master nodes in each zone are interconnected, which contributes to normal operation of the data center. For each of zones in the edge layer, there are multiple connections to the master nodes in the core layer. The whole network failure will not occur in case of the breakdown of certain master node thus the network connectivity can be guaranteed to a great extent.

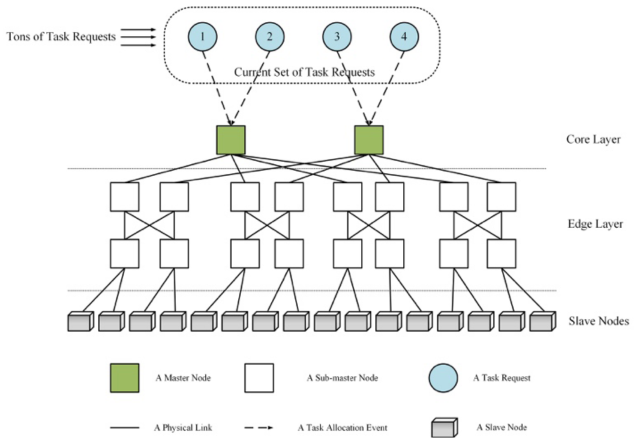


FIGURE 1. The proposed problem.

On the whole, with this pattern of fully connected network in this work, the overall throughput of system can be improved. The efficient parallel execution for big data processing can be achieved in virtue of using the multiple-branching tree construction. By using the cooperation of the core layer and edge layer, which consists of master nodes and sub-master nodes respectively, the requested tasks could be allocated into corresponding slave nodes successfully and that the reliable transmission of data can be guaranteed in case of certain processing infrastructure going down. Furthermore, taking advantage of the fat-tree based construction, we are able to achieve the load balancing of nodes and links in the case that tons of parallel tasks need to be addressed for big data.

As is well-known, numbers of physical nodes and links in this structure possess the capability of service for task requests. It is noting that different task allocation strategies may lead to different load distribution and cause different execution efficiency and external service capability. It is undoubtedly that an optimal large-scale tasks allocation approach should possess capabilities to achieve a proper task assignment, avert the overload of nodes and links, and thus improve the resource utilization rate of system. What's more, the overhead of computing in nodes and data transmission via links should be decreased while the overall energy consumption should be decreased as much as possible, which contributes to efficient task processing in the overall system. Thus it is necessary to design and implement a reasonable and efficient task allocation approach in the cloud environment.

IV. PRELIMINARIES

A substrate network can be denoted as a weighted undirected graph $G_{sub} = (N_{sub}, L_{sub}, A^n_{sub}, A^l_{sub})$, N_{sub} represents the set of related physical nodes, and L_{sub} represents the set of physical links in the substrate network. n_{sub} is an element of N_{sub} ($n_{sub} \in N_{sub}$), and l_{sub} is an element of L_{sub} ($l_{sub} \in L_{sub}$). n_{sub} and l_{sub} are associated with attributes A^n_{sub} and A^l_{sub}. In this paper, the attributes of a substrate network are usually the available CPU capability ($CPU(n_{sub})$) and network bandwidth ($BW(l_{sub})$).

Similar to the substrate network, a virtual network request can also be denoted as a weighted undirected graph $G_{vir} = (N_{vir}, L_{vir}, R^n_{vir}, R^l_{vir})$, N_{vir} is the set of virtual nodes (n_{vir}s) ($n_{vir} \in N_{vir}$) and L_{vir} is the set of virtual links (l_{vir}s) ($l_{vir} \in L_{vir}$). n_{vir} and l_{vir} are associated with the resource constraint set R^n_{vir} and R^l_{vir}. Generally, R^n_{vir} consists of the CPU request ($CPU(n_{vir})$) of each virtual node and R^l_{vir} consists of bandwidth request ($BW(l_{vir})$) of each virtual link.

Based on the previous work [6], in this paper, the standard deviation of CPU utilization rate is employed to represent load balancing degree, ε. The load balancing effect can be reflected by the load balancing degree. U^i_{vir} represents the current CPU utilization rate of the ith available node and n represents the total number of available nodes. The formula for the average resource utilization rate at can be defined as:

$$Avg(U) = \sum_{i=1}^{n} U^i_{vir}/n. \tag{1}$$

Then the load balancing degree can be obtained, and the formula is as follows:

$$\varepsilon(G_{vir}) = \sqrt{\frac{1}{n}\sum_{n=1}^{n}(U^i_{vir} - Avg(U))^2}. \tag{2}$$

Hereupon, the problem of node energy conservation can be defined as two major aspects: the node energy consumption of virtual network mapping process itself and accomplishing parallel tasks of a virtual request. Based on previous work, we analyze and quantify node energy consumption of mapping process for accepting a virtual request in virtue of calculating the integral of electricity price and power loss in time. The node energy consumption ΔEn herein can be defined as follows:

$$\Delta En = \sum_{v\in N_{vir}} \sum_{n\in N_{sub}} x^v_n \Delta Pn^v_n \int_{t_i}^{t_j} \Pr_n(t)dt. \tag{3}$$

where x^v_n represents the current status of mapping a virtual node $v \in N_{vir}$ onto the physical node $n \in N_{sub}$. If the node v has been successfully mapped onto the node n, $x^v_n = 1$, or $x^v_n = 0$. ΔPn^v_n represents the additional energy consumption of mapping the virtual node v onto the physical node n. Aiming at accurately representing the electricity price of node n at time t, we construct a discrete time model $Pr_n(t)$ with the time window [24].

For the phase of accomplishing the parallel tasks of the virtual request accepted, we can define the node energy consumption in unit time of virtual network mapping process for accepting N_{vir} as follows:

$$E_{ut}(N_{vir}) = \sum_{n\in N_{sub}} \alpha_n(p_n - q_n) + \sum_{n\in N_{sub}} w_n CPU(n). \tag{4}$$

where α_n is defined to represent the basic node energy consumption in unit time under the work. p_n and q_n are defined as the operation status of the node n after accepting N_{vir} and before it respectively, and if the node n is under the work, their values are set to 1, or they are set as 0. w_n is defined as the power consumption of unit CPU utilization of node n in unit time. Here, we can define the node energy consumption for accepting N_{vir} during $T(N_{vir})$ which represents the lifetime of finishing the parallel tasks of a virtual request:

$$E\ (N_{vir}) = E_{ut}\ (N_{vir})^*\ T\ (N_{vir})\ . \tag{5}$$

In consideration of minimizing bandwidth cost, according to the given DAG of tasks, the bandwidth cost value of each link in the whole available given path can be defined as $BW(l_{vir})$. Assume that there are p available links in the whole path, and the sum of total bandwidth cost can be defined as follows:

$$C(L_{vir}) = \sum_{l_{vir}\in L_{vir}} \sum_{l_{sub}\in L_{sub}} BW(l_{vir}). \tag{6}$$

Similar to the problem of node energy conservation, the link energy conservation issue is also defined as two major aspects: the link energy consumption during virtual network mapping process and completing the parallel tasks for the accepted virtual request. In this work, we can define the former as follows:

$$\Delta El = \sum_{l_{vh}\in L_{vir}} \sum_{l_{no}\in L_{sub}} y^{vh}_{no} \Delta Pl^{vh}_{no} \int_{t_i}^{t_j} \Pr_n(t)dt. \tag{7}$$

where y^{vh}_{no} represents the current status of mapping the virtual link l_{vh} onto the physical link l_{no}. If the virtual link l_{vh} has been successfully mapped onto the physical link l_{no}, $y^{vh}_{no} = 1$, or $y^{vh}_{no} = 0$. We use ΔPl^{vh}_{no} to represent the energy consumption of mapping the the virtual link l_{vh} onto the physical link l_{no}.

Aiming at completing the parallel tasks of the virtual request accepted, the link energy consumption in unit time of virtual network mapping process for accepting L_{vir} can be defined as follows:

$$E_{ut}(L_{vir}) = \sum_{(n,o)\in L_{sub}} \beta_{no}(y_{no} - x_{no}). \tag{8}$$

where β_{no} represents the basic link energy consumption in unit time under working. y_{no} and x_{no} are defined as the operation state of the link no after accepting L_{vir} and before it respectively, and if the link no is under working, their values are set as 1, or they are set as 0. The link energy consumption of completing the parallel tasks after accepting L_{vir} during the lifetime $T(L_{vir})$ can be expressed as follows:

$$E(L_{vir}) = E_{ut}(L_{vir})^{*}T(L_{vir}). \tag{9}$$

We define the optimization problem in this work with the objective function of minimizing the overall load balancing degree, energy consumption and bandwidth cost for virtual network mapping in distributed environment. The constraints are also defined in detail as follows according to the objective conditions.

Optimization Problem:

Minimize:

$$f_1 = \sqrt{\frac{1}{n}\sum_{n=1}^{n}(U_{vir}^{i} - Avg(U))^2}. \tag{10}$$

$$f_2 = \sum_{v\in N_{vir}}\sum_{n\in N_{sub}} x_n^v \Delta Pn_n^v \int_{t_i}^{t_j} \Pr_n(t)dt + (\sum_{n\in N_{sub}} \alpha_n(p_n - q_n) + \sum_{n\in N_{sub}} w_n CPU(n))^{*}T\ (N_{vir}). \tag{11}$$

$$f_3 = \sum_{l_{vir}\in L_{vir}}\sum_{l_{sub}\in L_{sub}} BW(l_{vir}). \tag{12}$$

$$f_4 = \sum_{l_{vh}\in L_{vir}}\sum_{l_{no}\in L_{sub}} y_{no}^{vh} \Delta Pl_{no}^{vh} \int_{t_i}^{t_j} \Pr_n(t)dt + (\sum_{(n,o)\in L_{sub}} \beta_{no}(y_{no} - x_{no}))^{*}T(L_{vir}). \tag{13}$$

Subject to:

Resource capacity constraint:

$$\forall v \in N_{vir},\quad \forall n \in N_{sub},\ \begin{cases} x_n^v \times CPU(v) \le CPU(n) \\ x_n^v \times Dis(Loc(n), Loc(v)) \le D \end{cases} \tag{14}$$

$$\forall (n,o) \in L_{sub},\quad \forall (v,h) \in L_{vir}, y_{no}^{vh} \times BW(l_{vh}) \le BW(l_{no}) \tag{15}$$

Connectivity constraint:

$$\forall n \in N_{sub},\quad \forall (v,h) \in L_{vir},$$
$$\sum_{(n,o)\in L_{sub}} y_{no}^{vh} - \sum_{(o,n)\in L_{sub}} y_{on}^{vh} = \begin{cases} 1, & if\ x_n^v = 1 \\ 1, & if\ x_n^h = 1 \\ 0, & otherwise \end{cases} \tag{16}$$

Variable range constraint:

$$\forall n \in N_{sub},\quad \sum_{v\in N_{vir}} x_n^v \le 1$$
$$\forall v \in N_{vir},\quad \sum_{n\in N_{sub}} x_n^v = 1 \tag{17}$$
$$\forall n \in N_{sub},\quad \forall v \in N_{vir},\ x_n^v \in \{0,1\}$$
$$\forall (n,o) \in L_{sub},\quad \forall (v,h) \in L_{vir},\ y_{no}^{vh} \in \{0,1\} \tag{18}$$

V. THE PROPOSED SCHEME

Aiming at the efficient parallel execution of large-scale tasks for big data learning, a novel approach TOPE has been presented in this work, which is used to explore an optimal task allocation scheme to realize the overall load balancing, energy conservation and minimization of bandwidth cost in distributed environment. TOPE combines the distributed multi-objective PSO (Particle Swarm Optimization) and deep reinforcement learning herein. It has achieved massive tasks allocation in virtue of virtual network mapping. In order to explore the optimal virtual node mapping decision, an efficient virtual node mapping method based on the distributed multi-objective PSO has been proposed. Subsequently, we employ the Q-learning with deep neural network for the virtual link mapping decision. Taking advantage of the continual interactions with the surroundings and tryouts, Q-learning preferentially selects the better link mapping scheme. Eventually, the optimal virtual network mapping scheme is achieved. The large-scale tasks are allocated onto proper physical nodes for efficient parallel execution for big data.

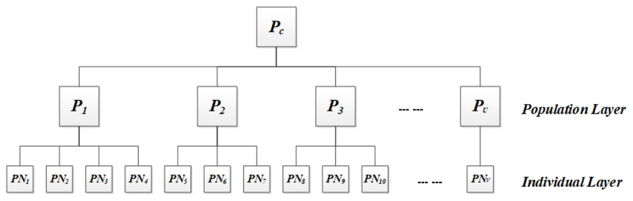


FIGURE 2. Architecture of the distributed multi-objective PSO method.

A. VIRTUAL NODE MAPPING SCHEME

Aiming at tackling the multi-objective problem of overall load balancing and the node energy conservation, different from previous work, we have proposed a virtual node mapping method based on distributed multi-objective PSO (Particle Swarm Optimization). Based on the idea of using distributed methods for large-scale tasks, V physical nodes in the substrate network are grouped into U sets, with each one corresponding to a PSO population for fitness evaluation. Fig. 2 describes the construction of the proposed distributed method. Similar to master-slave framework, V populations in the '*Population Layer*' can be implemented on parallel hardware, which are controlled by the master node P_c. P indicates a population as well as PN_i in the '*Individual Layer*' is the available physical nodes. The '*Individual Layer*' allocates the individual evaluations of each PSO population onto a corresponding set of slave physical nodes to maximize the benefits.

In respect of the velocity and position update of particles, taking advantage of the non-dominated sorting and the crowding degree comparison method, we can attain the global optimal solution which represents the optimal virtual node mapping scheme. In the meantime, the idea of non-uniform mutation is employed to improving the diversity of solutions and avoiding the local optimum, thus it accelerates the convergence of our approach.

Specifically, particles aggregate to the personal best position vector and global best position vector at a certain rate to iterate to search the optimal solution. Herein, each particle represents a potential solution of the proposed problem and it corresponds to a fitness value. Thus we need to define some critical parameters for the following optimization. The particle position vector $X_i = \left[x_i^1, x_i^2, \ldots, x_i^H\right]$ is defined as the *ith* possible mapping scheme. H represents the number of nodes in the virtual network. x_i^j is a positive integer and it represents the serial number of substrate node that the *jth* virtual node selects from the list of candidate substrate nodes. The particle velocity vector $V_i = \left[v_i^1, v_i^2, \ldots, v_i^H\right]$ is defined as the adjustment decision of mapping schemes which is employed to prompt the current mapping scheme to adjust to better one. v_i^j is a binary variable and in the case $v_i^j = 0$, the *jth* virtual node needs to reselect a node from substrate network for the virtual node mapping.

The complete method is illustrated as follows. First, we initialize the solution set. The values of objective function f_1 and f_2 of each virtual node mapping scheme can be calculated. For the multi-objective optimization problem of node load balancing and energy conservation, we can't ensure that each of the candidate optimal mapping schemes will bring sufficient performance revenue from parallel execution. Based on this fact, aiming at not only fully meeting the resource requests of virtual network but also realizing the overall load balancing and energy conservation, we define each node mapping scheme as personal best solution for the non-dominated sorting in the solution set. By this means, the solutions which are divided into c levels ($p_1, p_2, \ldots, p_c$) can be attained. Then we select the solutions belonging to the non-dominated solution set for crowding degree comparison. The solution with largest crowding degree is the current global best solution. The detailed process of non-dominated sorting and crowding degree comparison can be found in [9].

The process of exploring the global best solution is as follows. We define k as the number of iterations and the maximum number of iterations is set to K.

1. Based on the current personal best solutions and the global best solution, we can attain a node mapping scheme set $pop(k)$ according to the following velocity and position updating formulas:

$$V_{i+1} = wV_i + c_1 r_1 (X_{pb} - X_i) + c_2 r_2 (X_{gb} - X_i). \tag{19}$$

$$X_{i+1} = X_i + V_{i+1}. \tag{20}$$

where X_{gb} refers to the global best position vector and X_{pb} represents the personal best position vector. w is the inertia weight, c_1 and c_2 are learning factors. r_1 and r_2 are both random numbers which are uniformly distributed from 0 to 1.

2. The non-dominated comparison is carried out in each solution of node mapping scheme sets $pop(k)$ and $pop(k\text{-}1)$. We update the solutions dominated in $pop(k\text{-}1)$ to the better solutions in $pop(k)$. Thus the personal best node mapping scheme set $pbpop(k)$ is obtained.

3. The non-dominated sorting and crowding degree comparison are performed after combining $pop(k)$ and $pbpop(k)$. Thus we can get the $gbpop(k)$, and then we can get the global best solution.

4. The non-uniform mutation is introduced into $gbpop(k)$ for avoiding the local optimum. A probability is generated randomly herein. According to the probability, the global best solution is interchanged with any personal best solution. Detailed process can be found in [25].

5. Repeat the above steps *1-4*.

B. VIRTUAL LINK MAPPING SCHEME

In this work, we minimize the cost of link between each two adjacent nodes while satisfying the virtual requests. Based on this argument, we employ multi-agent reinforcement learning to implement virtual link mapping We can treat the whole network as a multi-agent system (MAS). Each node can be treated as an agent with independent learning capability. It is able to take the information about various aspects into account in case of selecting an adjacent node as the next hop node. We can describe the whole mathematical model in virtue of Markov decision process (MDP) herein. Moreover, we employ the distributed value function (DVF) based Q-learning for the virtual link mapping decisions in this work. The updating rule of DVF based Q-learning is as follows:

$$Q(s_i, a_i) \leftarrow (1-\alpha)Q_i(s_i, a_i) + \alpha(r_i(s_i, a_i) + \beta \sum_{j \in Nb(i)} \omega_i(j) V_j(s_j)). \tag{21}$$

where α is the learning rate and $r_i(s_i, a_i)$ represents the immediate cost that the state s_i receives from the environment by taking the action a_i. β represents a discount factor and $\omega_i(j)$ represents a weighted value. $V_j(s_j)$ is the value function of the adjacent node j of node i.

The specific steps of the MDP for virtual link mapping can be described as follows:

State Space. The state of node i can be defined as:

$$s_i^n = \left\{(b_{ij}^n, e_{ij}^n) \mid j \in Nb_i\right\}. \tag{22}$$

where b_{ij}^n represents the link quality between node i and its adjacent node j, and e_{ij}^n represents the queue size of adjacent node j. Nb_i is defined as the adjacent node set of node i. The set of all possible values for s_i^n can be defined as S_i

Action Set. For the action $a_i^n(s_i^n) \in A_i$ that the node i takes, it is only related to its own current state. And we can define A_i as follows:

$$A_i = \left\{sl_j, j \in Nb_i\right\}. \tag{23}$$

where sl_j refers to that the node i selects the node j as the next hop node.

Immediate Cost. The immediate cost is employed to reflect the current operation state and the efficiency of virtual link mapping. To minimize the objective functions f_3 and f_4, we introduce the following cost function. The node i with a failed mapping on the link between node i and j receives a cost K_{fail} . Otherwise, a cost is assigned to the node i based on its successful link mapping. Specifically, the cost of node i is defined as:

$$r_i^n(s_i^n, a_i^n(s_i^n)) = \begin{cases} K_{fail}, & \textit{mapping failed}; \\ K_1 \cdot f_3 + K_2 \cdot f_4, & \textit{success}. \end{cases} \quad (24)$$

where K_1 and K_2 are both positive weights. The cost function is suitable for minimizing f_3 and f_4 in a distributed way due to the fact that it takes bandwidth cost and consumed link energy into consideration.

Q-value Update. After obtaining the immediate cost of node i, we update the Q-value of an action a on state s in manner of DVF based Q-learning:

$$\begin{aligned} Q_i^{n+1}&(s_i^n, a_i^n(s_i^n)) \\ &= (1-\alpha)Q_i^n(s_i^n, a_i^n(s_i^n)) + \alpha(r_i^n(s_i^n, a_i^n(s_i^n)) \\ &\quad + \beta\omega(i,j) \min_{s_j^n \in S_j, a_j^n \in A_j} Q_j^n(s_j^n, a_j^n(s_j^n)) \\ &\quad + \beta \sum_{i' \in Nb_i, i' \neq j} \omega(i,i') \min_{s_{i'}^n \in S_{i'}, a_{i'}^n \in A_{i'}} Q_{i'}^n(s_{i'}^n, a_{i'}^n(s_{i'}^n))). \quad (25) \end{aligned}$$

where *ω(i,j)* refers to the weight of long-term cost that the node i receives from the selected node j. *ω(i,i')* is the weight of long-term cost that the node i receives from its adjacent node.

It is noting that "curse of dimensionality" may occur when MDP possesses the huge state space and action space. Based on this argument, the deep neural network is introduced to approximate the value function $Q(s, a)$.

The neural network consists of input layer, hidden layer and output layer, and the front layer and the back layer are connected by weights. The network model can turn to the back propagation in case of an error between the actual output and the expected output in the output layer. It adjusts the weights of each layer in right of gradient descent method to approach the minimum of output error (*OE*). For the model learning, the forward propagation and back propagation alternate until the error reaches an excepted range.

For the neural network based reinforcement learning process, at time t, the agent tasks action a_t and receives the immediate cost $r(s_t, a_t)$. Then the system inputs the current state-action pair (s_t, a_t) and the immediate cost $r(s_t, a_t)$ of MDP into the neural network. The neural network approximates the value function according to the input and immediate cost. And then it outputs the estimated value of the value function to the agent. The agent employs the estimated value to carry out the iteration of the value function. The weight vector adjustment can be achieved by the back propagation of the learning result $Q(s_t, a_t)$. The designed model no longer stores and updates the value function estimation table $Q(s, a)$. It only needs to store the weights of each neuron in the neural network. The storage scale is only related to the construction of the designed neural network model.

In order to avoid the interaction caused by simultaneous computation and training of Q-network, we have trained two neural networks with identical parameters, including the main network and the target network. The main network is used to update the weight in real time, while the target network keeps the weight unchanged temporarily. The weight of the main network is assigned to the target network after a period of time. It effectively avoids the instability caused by frequent weight updates.

The main steps of the deep Q-network are as follows:

1. Initialize the weights of the main network and make the values randomly distributed within [−1,1]. And then assign the weights of the main network to the target network. Initialize the experience-replay memory S_p to make its capacity reach the set number C.

2. The current environmental state s is detected for the tryout of model.

3. The selection of current action a should be carried out, which is determined by the method of ε-greed, where ε is the exploratory utilization rate. The probability of randomly selecting action a depends on ε. The maximal Q-value is used as the selection criterion to select the action value a. Generally, the value of ε is designed to decrease gradually. It is mainly because more different actions are needed to explore the impact to the environment in the initial stage, so as to avoid the overall system falling into the local optimum. The model is able to select the action with the maximal Q-value to achieve good results after a period of time. Thus we increase the probability of selecting the action values that have been learned before.

$$\varepsilon_x = 1 - \frac{x}{Y}. \quad (26)$$

where x is the current training step number and Y is the total step number at the initial setting.

4. The reward value r can be obtained. Store $< s, a, r, s' >$ into S_p with the updated state s'. Hereupon the initial sample will be replaced by the latest sample.

5. The sample $< s, a, r, s' >$ is randomly selected from S_p. The target Q-value z_j can be calculated as follows:

$$z_j = \begin{cases} r_j, & \text{if the episode ends at the state } s_{j+1}; \\ r_j + \gamma \max_{a'} Q(s', a', \omega'), & \text{otherwise.} \end{cases} \quad (27)$$

where Q(s', a', ω') is the Q-value of the target network. The parameters of the convolutional neural network are updated according to the gradient descent method. Herein, the loss function L_j is used for the parameter update of the Q-network. We sample in batches from S_p and update network parameters. The values of the main network weight are assigned to the target network weight every W steps.

$$L_j = \frac{1}{m} \sum_{j=1}^{m} (z_j - Q(s, a, \omega))^2. \quad (28)$$

6. Repeat the above steps*2-5*.

For the application of the deep network model, the corresponding action, namely the optimal virtual link mapping decision will be output when the current state including the link quality and the queue size of adjacent nodes is input.

The learning agent runs on the training data set for 2000 epochs. With the training going, the random sampling allows the learning agent to explore different possibilities. The learning agent may receive a good solution occasionally and a decent reward which helps the main network to learn to make better decisions. Herein we randomly sample 200 times from the experience-replay memory for training and update the gradient after each training.

In our neural network model, the first convolutional layer includes 32 convolution kernels. The size of each convolution kernel is 5×5. We set the size of stride to 1. The matrix size of max-pooling layer is 2×2 and its stride size is set to 2. The second convolutional layer includes 64 convolution kernels and other parameters are same as those of the first layer. Our training period often lasts 2-3 hours. What's more, the learning rate of model is set to 0.001. The optimization method Adam is utilized to achieve the dynamic updating of strides.

VI. DISCUSSION

In this paper, depending on the cooperation of master nodes and sub-master nodes in the fat-tree, each node can transmit data through other paths in case of certain central processing equipment going down. Meanwhile, it contributes to reduce the load of some links with less bandwidth and makes the topology more stable. The full connection pattern significantly increases the throughput of systems, thus it accelerates the information interaction and benefits to the raise of overall system performance.

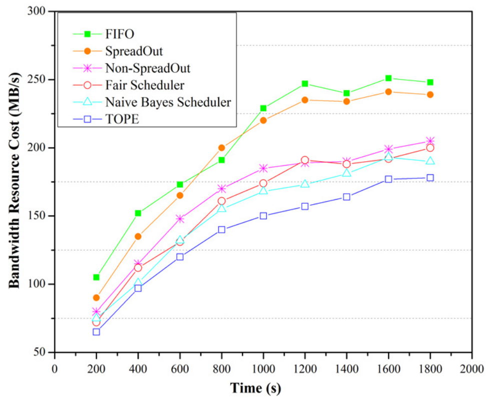
In order to improve the efficiency of big data processing, a two-phase optimization to parallel execution methodology has been proposed, and it mainly solves the proposed problem which is a NP-hard combination optimization problem. Generally, the previous work tends to ignore the resource requirement of nodes or links, omit the access control of virtual cluster requests and only take some special topology structures into consideration. Thus, it may result in high algorithm complexity, load imbalance, and overmuch bandwidth cost. Based on the above, different from the previous point of view, a "divide-and-conquer" strategy has been utilized. We have proposed a two-phase methodology TOPE to conduct the large-scale tasks processing instead of solving it all at once. Fundamentally, TOPE has ingeniously solved the combination optimization problem by a combination of a distributed method and deep reinforcement learning. Eventually, we have accomplished the optimization for the problem of load balancing, bandwidth cost minimization and energy conservation simultaneously.

During the virtual node mapping stage, based on the idea of using distributed methods for large-scale tasks, V physical nodes in the substrate network are grouped into U sets, with each one corresponding to a PSO population for fitness evaluation. Hereupon, we formalize each feasible solution (node mapping scheme) as a kinetic particle in the population. It explores and modifies the optimal position vector by continuously moving in the domain of definition. Herein, for our multi-objective combinatorial optimization, we have introduced the multi-objective PSO. By using the non-dominated sorting and the crowding degree comparison method, we can obtain the global optimal solution which represents the optimal virtual node mapping scheme. In the meantime, the idea of non-uniform mutation is employed to improving the diversity of solutions and avoiding the local optimum, thus it accelerates the convergence of our approach. The process of searching the optimal node scheme in TOPE is in line with idea of heuristic algorithm. It is mainly because that the solution obtained by the heuristic algorithm after each iteration search may not be the optimal solution. But taking advantage of unceasing searching and correcting the obtained solution in several iterations, the optimal solution can be gradually approached. And this process conforms the target of the proposed TOPE, the optimal scheme of node mapping can be obtained to achieve the overall load balancing and energy conversation eventually. It relieves the workload of some nodes with weaker processing capability. Furthermore, it improves the utilization rate of computing resource and the energy efficiency.

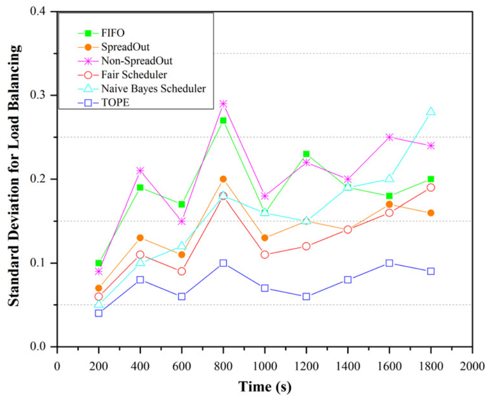
During the virtual link mapping phase, we have found that the issue of virtual link mapping can be transformed into the problem of optimal link selection. And it can be transformed into a kind of problem of MDP. Based on the multi-agent reinforcement learning theory, each node treated as an agent is able to employ the unceasing interactions with environment and tryouts to evaluate the feedback from the surroundings for optimizing the future decision-making. During the big data processing, it is noting that "curse of dimensionality" may occur when MDP possesses the huge state space and action space. Based on this argument, the deep neural network is introduced to approximate the value function Q. Besides, in order to avoid the interaction caused by simultaneous computation and training of Q-network, we have trained two neural networks with identical parameters, including the main network and the target network. The main network is used to update the weight in real time, while the target network keeps the weight unchanged temporarily. The weight of the main network is assigned to the target network after a period of time. The parameters of the neural network are updated by using gradient descent method through back propagation. One of the advantages in the designed model is to use experience-replay memory. It randomly and uniformly samples in experience-replay memory, the correlation among training samples is broken. At the same time, by averaging several samples in the past, it not only smooths the distribution of training samples, but also alleviates the problem of sample distribution variation. It effectively avoids the instability caused by frequent weight updates. Once the Q-value converges, the target value can be found, that is, the deterministic strategy has been obtained. To sum up, the optimal virtual link mapping decision with low bandwidth cost and energy consumption can be obtained.

TABLE 1. Comparison of the task request datasets.

Dataset	Main Features and Limitation
Synthetic Task Dataset	The workload is accumulated from the cloud data center within 15 days, including 35 batches of task requests containing 4250 requests with different resource requirements.
GoCJ	It is based on jobs size behaviors witnessed in the analysis of the Google cluster traces of 29 days, and the MapReduce logs from the M45 supercomputing cluster of 10 months. The GoCJ dataset is an alternative to the real workload on a Cloud.
Facebook Hadoop workload	The workload is based on Hadoop traces on 600 machine cluster on Facebook spans over 6 months duration from May 2009 to October 2009 containing 1 million jobs. The jobs in the workload are recorded with submit time and inter-job_submit_gap parameters.

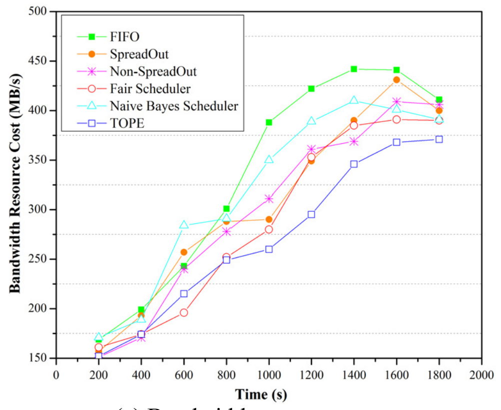


(a) Bandwidth resource cost.

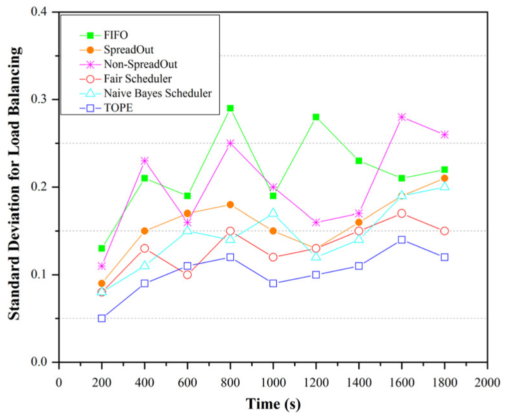


(b) Load balancing effect.

FIGURE 3. Performance comparison among different approaches on the synthetic task dataset with the task scale of 1250 task requests.



(a) Bandwidth resource cost.



(b) Load balancing effect.

FIGURE 4. Performance comparison among different approaches on GoCJ with the task scale of 1950 task requests.

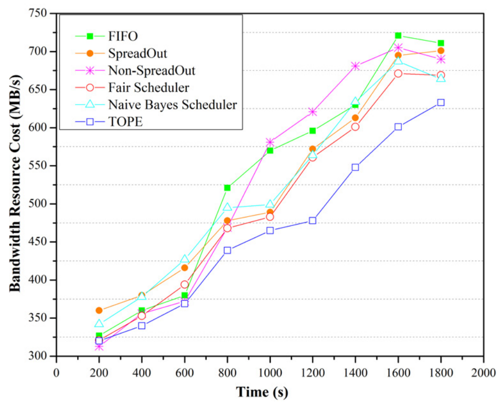
VII. EXPERIMENTS

In this section, we will compare TOPE with the existing approach FIFO, SpreadOut and Non-SpreadOut [26], Fair Scheduler [27] and Naïve Bayes Scheduler [28] through the following aspects: bandwidth resource cost, load balancing effect and energy consumption.

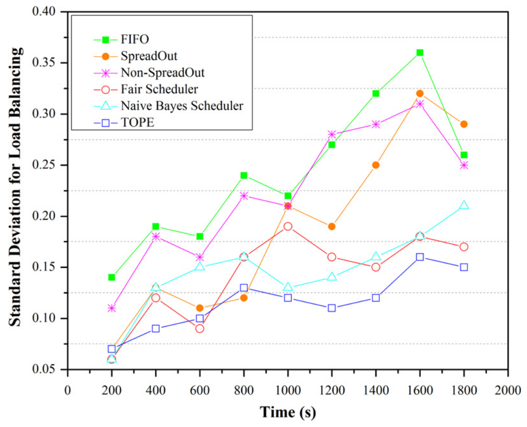
We exploit the OpenStack [29] to create virtualization scenarios for deployment of the big data processing framework Spark [30]. All the experiments are performed on the same Linux workstation with an Intel Xeon with three 3.4 GHz CPUs and 256GB memory. Herein 40 processing nodes with different configuration are created. Herein the latest version of Spark 2.3.0 is used, which needs to be built on the basis of Hadoop. And the Zookeeper is high-availability of the master. We use spark-2.3.0-bin-hadoop2.6.tgz since the Hadoop version is 2.6. As shown in Table 1, 35 batches of task requests containing 4250 requests with different resource requirements continuously getting to cloud data center compose the synthetic task dataset. We also conduct tests on Facebook Hadoop workload and Google Cloud Jobs Dataset (GoCJ) [31] to validate TOPE's capability on large-scale tasks processing.

As shown in Fig. 3(a), we can observe that TOPE always possesses less bandwidth cost on our synthetic task dataset. From the perspective of the bandwidth economy, the TOPE approach can decrease additional consumption of the bandwidth resource in the substrate network in the long run. In Fig. 3(b), it can be observed that with time changing, the standard deviation value of TOPE is always lower than those of other approaches, which indicates it possess better load balancing effect.

As shown in Fig. 4(a) and 4(b), TOPE always has less bandwidth cost and lower standard deviation values on GoCJ dataset. It indicates that our approach can maintain the



(a) Bandwidth resource cost.



(b) Load balancing effect.

FIGURE 5. Performance comparison on Facebook Hadoop workload with the task scale of 5894 task requests.

superiority though the task scale mushrooms in cloud environment. Fig. 5(a) and 5(b) show the performance of different approaches on Facebook Hadoop workload. It can be seen that TOPE still possesses the advantages of task processing in case of doubling the task scale. Although our approach doesn't perform better in comparison at a certain time, we still can conclude that TOPE can achieve comparatively better performance for large-scale tasks processing in the long term. It is mainly because that TOPE is able to simulate the particles in multi-objective PSO to take the global optimal position vector obtained from each iteration as the leader in the process of exploring the optimal solution. With the unceasing adjustment of velocity vector and position vector of each particle, the proper physical nodes for the corresponding virtual request can be selected from the substrate network in data center. It guarantees a high resource utilization of the system. The proposed TOPE approach considers not only the satisfaction for bandwidth request, but also the minimization of the total bandwidth cost. In the later stage, its growth rate of bandwidth cost gradually decreases. From the perspective of the bandwidth economy, the TOPE approach can decrease additional consumption of the bandwidth resource in the substrate network in the long run.

TABLE 2. Comparison in energy consumption on the synthetic task dataset, GoCJ with the task scale of 1950 task requests and Facebook Hadoop workload with the task scale of 6638 task requests. (E_1 and E_2 refer to node and link energy consumption during virtual network mapping process for accepting virtual request respectively; E_3 and E_4 represent node and link energy consumption of completing parallel tasks for accepted virtual requests respectively.).

Task Datasets	Methods	E_1	E_2	E_3	E_4
Synthetic Task Dataset	FIFO	373.53	147.62	2717.04	1249.51
	SpreadOut	351.91	128.79	2304.73	1001.40
	Non-SpreadOut	368.76	132.21	2579.47	1172.37
	Fair Scheduler	343.13	129.91	2285.03	1094.65
	Naïve Bayes	349.07	130.87	2374.55	1054.19
	TOPE	**237.81**	**91.57**	**2091.86**	**849.51**
GoCJ	FIFO	890.81	274.33	5983.01	3193.88
	SpreadOut	783.44	241.31	5551.68	**2834.41**
	Non-SpreadOut	812.52	267.95	5812.74	3113.63
	Fair Scheduler	712.77	251.34	5311.46	2991.86
	Naïve Bayes	731.69	254.19	5439.22	2901.18
	TOPE	**691.13**	**181.27**	**4835.09**	3007.61
Facebook Hadoop workload	FIFO	3099.64	1365.37	32187.71	21847.69
	SpreadOut	2677.04	1113.83	28793.57	18437.20
	Non-SpreadOut	3117.27	1234.11	30031.65	20071.72
	Fair Scheduler	2888.87	1201.45	27009.12	16774.37
	Naïve Bayes	2534.09	1119.27	25882.98	17993.83
	TOPE	**2268.66**	**999.15**	**24792.60**	**16373.96**

Aiming at validating the energy conservation capability of TOPE, we conduct three sets of experiments. Herein, we evaluate the energy consumption of virtual network mapping process and completing the parallel tasks for the accepted virtual request. We conduct several tests for different virtual requests and take the mean as the experimental result. The experimental results on different task datasets are illustrated in Table 2. It effectively demonstrates energy conservation capability of TOPE in large-scale tasks processing. SpreadOut and Fair Scheduler both possess low consumption, but TOPE attains much better results under the same condition. It is mainly because that our approach TOPE simultaneously takes load balancing, bandwidth cost and energy consumption into account via reasonable task allocation while satisfying virtual requests, the excess energy consumption of nodes and links is decreased to a great extent. The energy efficiency of whole system can be achieved.

VIII. CONCLUSION

This paper has proposed a novel approach TOPE, which aims at exploring the optimal task allocation scheme for big data processing. Herein, 1) a two-phase optimization methodology for large-scale task processing in big data environment is proposed; 2) the critical section of task processing problem in cloud computing is formalized by using multi-agent reinforcement learning. The deep reinforcement learning model is trained using deep neural network to approximate the value function, save parameters of the deep network model instead of state-action values as well as make reasonable virtual link mapping decisions. The virtual node mapping is achieved by designed distributed multi-objective swarm intelligence to realize the two-phase optimization for task allocation in the introduced fat-tree structure; 3) the superiority of TOPE for large-scale tasks processing is evaluated in comparison with state-of-the-art approaches in cloud environment. Experimental results show that TOPE can minimize bandwidth resource cost, realize overall load balancing and decrease energy consumption simultaneously.

In the future, it is necessary to increase the number of algorithm iterations in course of searching the optimal task allocation scheme, since the selection of maximum number of iterations is an open problem. A larger scale of experiments on TOPE is also one of our next research work. Moreover, the large-scale tasks processing more deeply combining with deep learning and reinforcement learning for big data is an important research direction in our following work.

REFERENCES

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[2] R. Ranjan, L. Wang, A. Y. Zomaya, D. Georgakopoulos, X.-H. Sun, and G. Wang, "Recent advances in autonomic provisioning of big data applications on clouds," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 101–104, Jun. 2015.

[3] S. Sagiroglu and D. Sinanc, "Big data: A review," in *Proc. Int. Conf. Collaboration Technol. Syst.*, San Diego, CA, USA, May 2013, pp. 42–47.

[4] J. Kempf, Y. Zhang, R. Mishra, and N. Beheshti, "Zeppelin—A third generation data center network virtualization technology based on SDN and MPLS," in *Proc. IEEE 2nd Int. Conf. Cloud Netw.*, Nov. 2013, pp. 1–9.

[5] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 20–27, Mar./Apr. 2013.

[6] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 305–316, Feb. 2016.

[7] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM Workshop Virtualized Infrastruct. Syst. Archit.*, Aug. 2009, pp. 81–88.

[8] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[9] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[10] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 4157–4162, Oct. 2010.

[11] F. Chaix, I. Fujiwara, and M. Koibuchi, "Suitability of the random topology for HPC applications," in *Proc. 24th Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Feb. 2016, pp. 301–304.

[12] L. Wang and G. Lu, "The dynamic sub-topology load balancing algorithm for data center networks," in *Proc. Int. Conf. Inf. Netw.*, Jan. 2016, pp. 268–273.

[13] X. Jin, H. Li, Y. Liu, and Y. Fan, "A study on load balancing techniques for task allocation in big data processing," in *Proc. Int. Forum Mech., Control Automat.*, 2016, pp. 212–218.

[14] T. Hirai, H. Masuyama, S. Kasahara, and Y. Takahashi, "Performance analysis of large-scale parallel-distributed processing with backup tasks for cloud computing," *J. Ind. Manage. Optim.*, vol. 10, no. 1, pp. 113–129, 2017.

[15] J. Gu, X. Lan, Y. Hao, and Y. Hu, "Arana: A cross-domain workflow scheduling system," in *Proc. 3rd Int. Conf. Wireless Commun. Sensor Netw. Adv. Comput. Sci. Res.*, 2016, p. 44.

[16] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and repurchasing for big data processing in multi-clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 266–277, Apr./Jun. 2016.

[17] V. Vashishth, A. Chhabra, and A. Sood, "A predictive approach to task scheduling for Big Data in cloud environments using classification algorithms," in *Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng.*, Jan. 2017, pp. 188–192.

[18] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, Apr. 2017.

[19] L. Huang, J. Li, J. Li, and D. Yi, "Computation partitioning for mobile cloud computing in a big data environment," in *Proc. 2nd Int. Conf. Frontiers Sensors Technol.*, Apr. 2017, pp. 312–316.

[20] C. Xu, K. Wang, P. Li, R. Xia, S. Guo, and M. Guo, "Renewable energy-aware big data analytics in geo-distributed data centers with reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, to be published.

[21] H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng, and A. Bouguettaya, "Adaptive and dynamic service composition via multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Web Services*, Anchorage, AK, USA, Jun./Jul. 2014, pp. 447–454.

[22] F. Bu, "An intelligent efficient scheduling algorithm for big data in communication systems," *Int. J. Commun. Syst.*, vol. 31, no. 16, 2017, Art. no. e03465.

[23] Y. He, C. Liang, F. R. Yu, N. Zhao, and H. Yin, "Optimization of cache-enabled opportunistic interference alignment wireless networks: A big data deep reinforcement learning approach," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, May 2017, pp. 1–6.

[24] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed Internet data centers in a multi-electricity-market environment," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.

[25] X. Zhao, X.-S. Gao, and Z.-C. Hu, "Evolutionary programming based on non-uniform mutation," *Appl. Math. Comput.*, vol. 192, no. 1, pp. 1–11, 2007.

[26] D. Cheng, X. Zhou, P. Lama, J. Wu, and C. Jiang, "Cross-platform resource scheduling for spark and MapReduce on YARN," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1341–1353, Aug. 2017.

[27] H. Chen and F. Z. Wang, "Spark on entropy: A reliable and efficient scheduler for low-latency parallel jobs in heterogeneous cloud," in *Proc. IEEE 40th Local Comput. Netw. Conf. Workshops (LCN Workshops)*, Oct. 2015, pp. 708–713.

[28] J. Dhok and V. Varma, "Using pattern classification for task assignment in MapReduce," in *Proc. ISEC*, 2010, pp. 1–10.

[29] H. Li, H. Zhou, H. Zhang, and B. Feng, "EmuStack: An OpenStack-based DTN network emulation platform," in *Proc. Int. Conf. Netw. Netw. Appl.*, Jul. 2016, pp. 387–392.

[30] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[31] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, p. 38, 2018.

YONGYI CHENG was born in 1982. He received the M.S. degree from the College of Computer Science and Technology, Jilin University, in 2006, where he is currently pursuing the Ph.D. degree. His major research interests include cloud computing, big data, and data mining.

GAOCHAO XU received the B.S., M.S., and Ph.D. degrees from the College of Computer Science and Technology, Jilin University, China, in 1988, 1991, and 1995, respectively, where he is currently a Professor and a Ph.D. Supervisor. His main research interests include distributed systems, grid computing, cloud computing, the Internet of Things, information security, software testing, and software reliability assessment. As a person in charge or a principal participant, he has finished more than ten national, provincial, and ministerial level research projects of China.

• • •