Multi-Dimensional Resource Allocation in Distributed Data Centers Using Deep Reinforcement Learning

Wenting Wei, *Member, IEEE*, Huaxi Gu, Kun Wang, Jianjia Li, Xuan Zhang, and Ning Wang, *Senior Member, IEEE*

***Abstract*—With the development of edge-cloud computing technologies, distributed data centers (DCs) have been extensively deployed across the global Internet. Since different users/applications have heterogeneous requirements on specific types of ICT resources in distributed DCs, how to optimize such heterogeneous resources under dynamic and even uncertain environments becomes a challenging issue. Traditional approaches are not able to provide effective solutions for multi-dimensional resource allocation that involves the balanced utilization across different resource types in distributed DC environments. This paper presents a reinforcement learning based approach for multi-dimensional resource allocation (termed as NESRL-MRM) that is able to achieve balanced utilization and availability of resources in dynamic environments. To train NESRL-MRM's agent with sufficiently quick wall-clock time but without the loss of exploration diversity in the search space, a natural evolution strategy (NES) is employed to approximate the gradient of the reward function. To realistically evaluate the performance of NESRL-MRM, our simulation evaluations are based on real-world workload traces from Amazon EC2 and Google datacenters. Our results show that NESRL-MRM is able to achieve significant improvement over the existing approaches in balancing the utilization of multi-dimensional DC resources, which leads to substantially reduced blocking probability of future incoming workload demands.**

***Index Terms*—Distributed data centers, multi-dimensional resources allocation, workload placement, balanced resource utilization, deep reinforcement learning, natural evolution strategy.**

I. Introduction

RECENT advances in cloud computing has given rise to extensive deployment of data centers (DCs). Thanks to virtualization techniques, cloud services offered by cloud providers such as Amazon, Google and Alibaba can be implemented on multiple distributed data centers for on-demand sharing of resources (CPU, memory, storage) [1]. Workload placement is a key function of resource management that is provided by Virtualization Infrastructure Managers (VIMs) over distributed data centers [2] for Web applications, enterprise IT infrastructures and network function virtualization (NFV) [3]. It is worth noting that it is possible to run multiple virtual machines on the same DCs, while an application (such as MapReduce and Spark) will be executed across multiple distributed DCs in parallel. Therefore, efficient resources managements in distributed DCs have attracted renewed attention in both industry and academia.

Although virtualization technology enables cloud service providers to expand the service scale and optimize server utilization at a lower cost, the distributed nature and coexistence of multidimensional resources over distributed DCs [4] pose many challenges in efficient utilization of resources and satisfaction of the SLA requirements from diversified applications. More specifically, different applications have diverse resource requirements, such as computation intensive, memory intensive and I/O intensive applications [5]. Improper resource allocations without consideration of balanced utilization of multidimensional resource may lead to resource fragmentations, where one type of resource will be exhausted while others are still sufficient at DCs. As a result, the waste of resources will occur due to resource fragmentation across DCs. Even, dynamic arrival or departure of workloads from diverse applications aggravates multi-dimensional resource fragmentations, if the balanced utilization of multi-dimensional recourses is not taken into account. Therefore, we focus on the essential and promising issue of resource allocation related to balanced utilization of multiple dimensions resources (i.e., CPU, memory, storage) over distributed DCs.

Manuscript received 1 February 2022; revised 21 June 2022 and 3 October 2022; accepted 5 October 2022. Date of publication 11 October 2022; date of current version 6 July 2023. This work was supported in part by the National Key R&D Program of China under Grant 2018YFE0202800, National Natural Science Foundation of China under Grant 62102302 and 61934002, the Natural Science Foundation of Shaanxi Province for Distinguished Young Scholars under Grant no 2020JC-26, the Science and Technology on Communication Network Laboratory under Grant BAX20641X008 and SXX21641X033, National Key R&D Program of China (2019YFE0113200) and National Natural Science Foundation of China (61901317). This work was also supported by The Youth Innovation Team of Shaanxi Universities. The associate editor coordinating the review of this article and approving it for publication was G. Casale. *(Corresponding author: Huaxi Gu.)*

Wenting Wei is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China, and also with the Science and Technology on Communication Networks Laboratory, China Electronics Technology Group Corporation, Shijiazhuang 050081, China (e-mail: wtwei@xidian.edu.cn).

Huaxi Gu, Jianjia Li, and Xuan Zhang are with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China (e-mail: hxgu@xidian.edu.cn).

Kun Wang is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China.

Ning Wang is with the Department of Electrical and Electronic Engineering, University of Surrey, GU2 7XH Guildford, U.K.

Digital Object Identifier 10.1109/TNSM.2022.3213575

Traditional resource allocation solutions in distributed DCs are mostly based on heuristics with the priori knowledge, which assume system environments and resource requirements can be well modeled with handcrafted parameters. Indeed, heuristics work well in certain domains, but often fail at dynamic and heterogeneous requirements from various applications, due to the lack of interaction with dynamic environment. Considering time-varying requirements of multi-dimensional resources in distributed DCs, designing a resource allocation scheme to meet stringent performance requirements without sacrificing the efficient usage of resources is still a challenge. Thanks to recent advances of artificial intelligence (AI) technologies, researchers have explored whether machine learning can provide an alternative solution for network management [6], [7], [8], [9], [10], [11].

Still, state-of-the-art self-adaptive resource allocation frameworks lack support for the growing diverse requirements from the emerging applications, in which balanced resource utilization is a critical option. Inspired by previous researches, we are motivated to further explore the balanced and efficient utilization of multi-dimensional recourses over distributed data centers using deep reinforcement learning (DRL). DRL is regarded as a worthwhile approach indeed, but is still a tentative exploration as current research has seldom focused on studying how to help agents of DRL make such much faster and diverse exploration for DRL-based resource allocation frameworks.

Therefore, in this work, we focus on multi-dimensional resource allocation across distributed DCs, where job requests arrive and depart dynamically in an online manner and resource requirements are not known beforehand. We propose a multi-resources manager with natural evolution strategy based DRL, termed as NESRL-MRM. Specifically, its agent pursues to balance the utilization of resources along multiple dimensions and maximize remaining available resource, which leads to accommodate future incoming jobs. To train such a DRL agent in the most efficient manner possible, we design the training process with NES strategy to train the agent NESRL-MRM, which approximates the gradient of the reward function in the parameter space by population-based evolutionary search. As a result, NESRL-MRM facilitates the dynamic allocation of multiple dimensional resources and promotes its adaptation to the dynamic changes of traffic and the differentiation of service requirements, so as to achieve the goal of dynamic, flexible and real-time resource allocation and management.

The main contributions of this paper are summarized as follows:

- We propose a DRL-based multi-dimensional resource allocation scheme across distributed data centers, termed as NESRL-MRM, whose agent pursues to balance multiple dimensional resources utilization and maximize remaining available resources, then in order to accommodate as many as possible future incoming workloads.
- In reward design of NESRL-MRM, to reduce the resource waste caused by multidimensional resource fragmentation (overloaded in some resources but underutilized in others), we define multi-dimensional resource balance index (MRBI) to quantify the level of balanced utilization of multi-dimensional resources, which can help in providing more opportunity to reserve underlying workload for future demands.
- To train such an DRL agent with much faster wall-clock time and diverse exploration, we design a natural evolution strategy (NES) based training algorithm to approximate the optimal action-value function. Without any domain-specific rule-based heuristic, the agent of NESRL-MRM effectively learns to make better selections by trial and error in the form of reward signals.
- To evaluate the effectiveness of NESRL-MRM, our simulations are driven by real-world data traces of well-known cloud providers (Amazon and Google). Extensive simulation results show that NESRL-MRM achieves a significant improvement over the existing approaches in balancing the utilization of multi-dimensional resources and reducing the blocking of requirements.

The remainder of this paper is organized as follows. We discuss related works in Section II and present the system model, the mathematical model and motivate our work by using a simple example in Section III. The NESRL-MRM is introduced with RL-based formulation and its training algorithm in Section IV. We present our simulation setup and evaluate the proposal compared with baseline schemes in Section V. Finally, the conclusions are drawn in Section VI.

II. Related Work

With the growth of cloud service, increasingly large amounts of data are computed and stored in distributed DCs. Thus, resource management over distributed DCs has drawn much attention and multiple research efforts have been pursued from many different perspectives.

In [12], authors formulated the task scheduling over geo-distributed DCs as an integer linear programming problem with the goal of shortening job completion times and cutting network costs. To achieve energy-efficient workload placement for DCs, authors in [13] constructed a rack-level power model that mapped the workload directly to its power dissipation and pursued an optimal workload allocation with minimized power consumption. Authors in [14] presented a virtual data center resource manager that exploited the network and topology knowledge, since they assumed traffic and topology were informed before so that the overall performance would be improved drastically. HUG [15] proposed a coflow scheduling algorithm to achieve multiresource fairness and maximize network utilization without sacrificing strategy-proofness. Similarly, aiming at achieving max-min fairness across jobs sharing these datacenters and improving job completion times, authors [16] designed a fair job scheduler to assign tasks belonging to multiple jobs across datacenters, which was formulated as a multi-objective problem and solved by transforming it into its single-objective subproblems.

The above works deal with resource management or job scheduling over distributed DCs, but pay no attention to the heterogeneity of both server specifications and demand profiles, which leads to a waste of resources due to unbalanced

utilization among different dimensional resources. Toward multiple and heterogeneous nature of resources, a few recent works focus on multiresource allocation in cloud computing.

Aiming at mitigating over-allocation and reducing resource fragmentation under multi-type resource scenario, a heuristic-based multi-dimensional resource scheduler (named Tetris) [17] was presented. Nevertheless, Tetris ignored the temporal resource usage changes during runtime and was designed under assumption of priori knowledge on both the resource requirements of tasks and resource availability at machines. Authors in [18] considered balanced usage of multiple dimensional resources to design a threshold-based VM placement strategy Min-DIFF, which is intended to balance the usage of resources and reduce the risk of PM overloading in an online manner but the threshold of each dimensional resource didn't vary with time-varying recourse requirements and resource usage.

Due to the variety of applications and dynamic changes of workload, traditional heuristic-based resource allocation schemes cannot solve online management problem effectively. Since the resource allocation across distributed DCs is time dependent planning problem and the impacts of allocation decisions are not immediately observable, a handcrafted heuristic would be sub-optimal, even doesn't work to handles time-varying resource usage aspects of the workloads. Inspired by the rapid growth of AI technologies, researchers have explored whether machine learning can provide an alternative solution for network traffic classification [19], network management [6], [7], [9], protocol designs [20], [21], [22] or performance modeling [23], and several works have attempted to optimize resource allocation using machine learning.

Aiming at the conflict between cloud service providers, DQN-based resource scheduling was proposed [10] for the optimal trade-off between energy consumption and task makespan. Focus on both resource allocation and power management in cloud computing systems, a RL-based hierarchical framework was proposed in [24], which was constituted by a global and local resource allocation and power management for VMs, respectively. Authors in [25] proposed an architecture of intelligent cloud resource management with deep reinforcement learning, and only gave a simple DRL-based architecture for online resource management to achieve cost reduction. To handle directed acyclic graph tasks in a cloud computing environment, a DRL-based online scheduling algorithm using deep Q-learning training method was presented in [26], whose reward principle was only simply designed according to busy and idle states of VMs. A DRL-based fair resource sharing method named FairTS was presented in [27], where the agent learned to shorten the average task slowdown while ensuring multi-dimensional fairness among the tasks. Focus on online multi-dimensional scheduling, authors presented a deep reinforcement learning (DRL) scheme termed as DeepRM [28]. To facilitate such large-scale RL tasks, deep neural networks (DNN) were attempted to an approximation function and the resource profiles were translated into distinct images as the input of DNN. More similarly, A2CScheduler was presented in [29] as a DRL algorithm with A2C to address the customized job scheduling problem in data centers, which designed a reward function related to averaged job waiting time, but did not cover balanced utilization of multiple dimensional resources.

Although the above works gives us an inspiration, still, state-of-the-art self-adaptive resource allocation frameworks do not meet the growing diverse requirements from the emerging applications, due to their absence of balanced resource utilization and resource fragmentation. Recently, related work TVW-RL [30] presented a DRL-based workload scheduler by considering both temporal resource-usage characteristics and various equivalence classes of time-varying workloads. The main innovation was elaborated its reward function to optimize utilization, fragmentation and resource exhaustion. But TVW-RL neither focus on how to measure the balanced resource utilization to alleviate resource fragmentation, nor do explore how to help such a complex agent of DRL make much faster and diverse exploration for DRL-based resource allocation frameworks.

Inspired by but different from previous works, we are motivated to further explore DRL-based online resource allocation optimization related to balanced and efficient utilization of multi-dimensional recourses over distributed DCs, while exploring how to quantify balanced utilization of multi-dimensional resources.

III. Problem Statement and Formulation

A. System Modeling

We consider a dynamic multi-dimension resource optimization across all DCs without knowledge about future incoming demands where multiple DCs are distributed. Multiple candidate DCs are present in clusters where workload may be placed. Each DC possesses three dimensions of resource space, including computing, storage and memory resources under the control of a local orchestration/control function. The workload may be an aggregation with requirements of CPU, storage and memory resources. We consider the model that job requests arrive and depart dynamically in an online manner, where resource requirements are not known beforehand. This dynamic nature produces a realistic scenario.

We transform such a dynamic resource allocation into an online optimization problem. The mathematical description of the proposed method is as follows. Assume that there are several distributed DCs, and the set of DCs is denoted as $\mathbb{D}$. The available resource space and the resource capacities of distributed DCs are represented as $\mathbb{AR} = (\mathbb{AR}^{CPU}, \mathbb{AR}^{Mem}, \mathbb{AR}^{Disk})$ and $\mathbb{CR} = (\mathbb{CR}^{CPU}, \mathbb{CR}^{Mem}, \mathbb{CR}^{Disk})$, respectively. The three dimensions of resource requirements from a workload are denoted as $\boldsymbol{\mathcal{QR}}_t = \mathcal{QR}^{CPU}, \mathcal{QR}Mem, \mathcal{QR}^{Disk})$.

B. Motivation Example

The main challenge of this scenario is the joint optimization of multidimensional resources. Here, we provide an example of unbalanced allocation of multidimensional resources. As observed from Fig. 1, there are two DCs in the running state. One data center (marked DC1) has provisioned 90% of

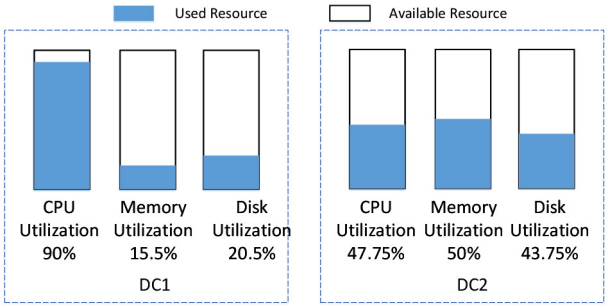


Fig. 1. A motivation example for balanced utilization of multi-dimensional resources.

its main CPU, 15.5% of its memory, and 20.5% of its disk for the already allocated workloads, whose multi-dimensional resources were allocated imbalancedly. The resources usage of other data center (marked DC2) is balanced, specifically, it is occupied by 47.75%, 50%, and 43.75% of its three dimensions of resources respectively. Under such a condition for DC 1, the allocation of new workload will be very difficult due to the low available CPU, but a high percentage of the memory and disk remain unused.

As shown in Fig. 2(a), assuming there are two new workload requirements (marked R1 and R2) that need to be allocated at the same time. Specifically, R1 requires 30%, 5%, and 10% of the total capacity for each dimension in multiple resource space, and R2 requires 5%, 15.5%, and 7.4% respectively. Meanwhile, there are two DCs in the running state in Fig. 2(b). One DC (marked DC1) has used 50% of its main CPU, 20% of its memory, and 30% of its disk for the already allocated workloads. The resource usage of the three dimensions of the other DC (marked DC2) is 40%, 65%, and 50%.

Under such a condition, Fig. 2(c) shows a sketch of multi-dimensional resources allocations by a traditional multidimensional resource allocation strategy Tetris [17]. Tetris firstly computes the alignment that is dot product between workload requirements and available resources of DC1, and the workload requirement with a larger alignment (which is R1) will be allocated to DC1 firstly. Then, it will compute the alignment between the left requirement (R2) and the updated available resources of the two DCs. Subsequently, the DC corresponding to the higher result (still DC1) will be chosen. As a result, the final utilization along three dimensional resources is unbalanced, and the allocation of new workload will be very difficult due to the low available CPU, but a high percentage of memory will remain unused. If a new workload requires 12.5%, 12.5%, and 7% of the total capacity for each dimension in multiple resource space respectively, unfortunately, there is no appropriate resource for DC1 to provide it despite redundant of Memory and Disk respectively. If multidimensional resource allocation considers balanced multidimensional resource utilization is executed, such as in the example in Fig. 2(d), a new workload with 12.5%, 12.5%, and 7% requirements for three dimensional resources will easily be placed in DC1. We can observe from the example that allocating resources in the proper DC will lead to a more balanced system and convenience in the subsequent placement of workload.

C. Mathematical Model

Our goal is to reduce the waste of multidimensional resources and improve resource utilization without knowledge about future incoming demands in such an online process. To facilitate the formulation of effective decisions that can help in providing more opportunity to reserve underlying workload for future demands, objectives can be transformed into achieving the maximum the minimum (bottleneck) available resource and maximizing the balanced usage of multidimensional resources across all data centers.

Multi-dimension resource balance objective: Balanced resource utilization facilitates multi-dimensional resource allocation in distributed DCs. Otherwise, unbalanced resource utilization (overloaded in some resources but underutilized in others) may cause a waste of resources, since it requires more extra DCs to accommodate workloads due to resource fragmentations [2].

Definition 1 [Multi-Dimensional Resource Balance Index (MRBI)]: is defined to quantify the balanced degree of multi-dimensional resource utilization. *MRBI* at any time slot t can be calculated as:

$$\mathcal{MB}_t = \min\left\{ \sqrt[n]{\prod_{i=1}^{n} \frac{\mathcal{AR}_{t,i}}{\mathcal{CR}_i}} \times \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \frac{\mathcal{AR}_{t,i}}{\mathcal{CR}_i}} \Big|_{i=1,2,3} \right\} \quad (1)$$

where $\mathcal{AR}_{t,i}$ is the residual available amount of the *i-th* dimensional resource at time slot t. $\mathcal{CR}_i$ denotes the capability of the *ith* dimensional resource. The ratio of $\mathcal{AR}_{t,i}$ to $\mathcal{CR}_i$ is called the normalized residual available resource. Note that we define *MRBI* according to the well-known Geometric-Arithmetic Mean (AM-GM) Inequality [31], [32]. The Geometric-Arithmetic Mean (AM-GM) Inequality states that the arithmetic mean of non-negative real numbers is greater than or equal to the geometric mean of set of data values. Further, equality holds if and only if every normalized residual available resource is the same. Therefore, the value of *MRBI* is between 0 and 1, and it is infinitely close to 1 when the normalized residual resources of every dimension is the same.

According to Eq. (1), *MRBI* is proportional to the geometric mean of the normalized 3-dimensional residual resources and inversely proportional to arithmetic mean of the normalized 3-dimensional residual resources. The larger *MRBI* is, the better balanced resource utilization is.

Available resources objective: We term the smallest value among the three dimensions of the available resources space as the minimum available multi-dimension resource for a single data center. The *minimum available multi-dimension resource* at any time slot t is denoted as Eq. (2).

$$\mathcal{MA}_t = \min\left\{ \frac{\mathcal{AR}_{t,i}}{\mathcal{CR}_i} \Big|_{i=1,2,3} \right\} \quad (2)$$

(a) Workload Requirements

(b) Original Resource Utilizations

(c) Allocation profiles with Tetris

(d) Allocation profiles with the balanced utilization

Fig. 2. A sketch of multi-dimensional resources allocations.

According to the mathematical model, optimization objectives are translated into maximizing the minimum (bottleneck) available multi-dimensional and multi-dimensional balance index, so as to reduce the waste of multidimensional resources caused by resource fragment while improving resource utilization. For any date center $D^l \in \mathbb{D}$, the optimization objective is listed as follows.

$$\textbf{Maximize}\left(min\left(\mathcal{MA}\left(D^l\right)\right), min\left(\mathcal{MB}\left(D^l\right)\right)\right) \quad (3)$$

Note that $\mathcal{MA}$ and $\mathcal{MB}$ are independent with each other and their weights can be adjusted according policies by cloud infrastructure providers.

It is worth noting that the focus of this paper is the optimization of multi-dimensional resources across distributed DCs, so the traffic demand optimization across inter-DC links is outside our scope.

IV. Design of NESRL-MRM: A DRL-Based Online Allocation of Multi-Dimensional Resource

To achieve the objectives above, we design an online multi-dimension resource manager with neuroevolution based DRL, termed as NESRL-MRM. In the design of NESRL-MRM, state-selection mapping policies are presented as a deep neural network that maps a raw observation to a selection, which is trained by a natural evolution strategy with some customizations, as detailed in the Training Design part. NESRL-MRM takes advantage of reinforcement learning to interact with the network environment and learn a resource allocation policy, aiming at balancing multi-dimension resource utilization and reducing resource fragmentation efficiently in distributed data centers.

A. Reinforcement Learning Formulation

According to the basic framework of reinforcement learning, a DRL-based multi-dimension resource management algorithm is designed in this section. In a reinforcement learning framework, it is assumed that an agent exists in the environment. A schematic diagram of our approach is shown in Fig. 3, where an agent interacts with its environment through observations and feedback. Specifically, the agent of NESRL-MRM observes a set of metrics including past resources allocation decisions and their reward feedbacks, and several raw signals (distributed data centers, currently jobs and their resource requirements, multi-dimensional resource utilization), which are incorporated into the state space at the *t-th* time slot (marked as s_t). These metrics are fed into a deep neural network that provides a scalable and expressive method for selections of candidate policy, and then the agent chooses a corresponding action a_t. Following the action, the state of the environment should be transferred to the next state s_{t+1} and the reward information r_{t+1} is observed and fed back into the agent. The agent uses the reward information to train and improve the neural network model, whose objective in the learning process is to maximize the expected cumulative discounted reward, i.e., the long-term total reward, by trial and error instead of the present reward.

The elements involved in this DRL-based design are as follows.

1) State Space: The state space is denoted as $\mathbb{S}$ and contains two elements that describe the current state.

IT resources requirement from dynamic arriving jobs, including the CPU, memory, and disk requirements that are noted as $\boldsymbol{QR} = (\mathcal{QR}^{CPU}, \mathcal{QR}^{Mem}, \mathcal{QR}^{Disk})$.

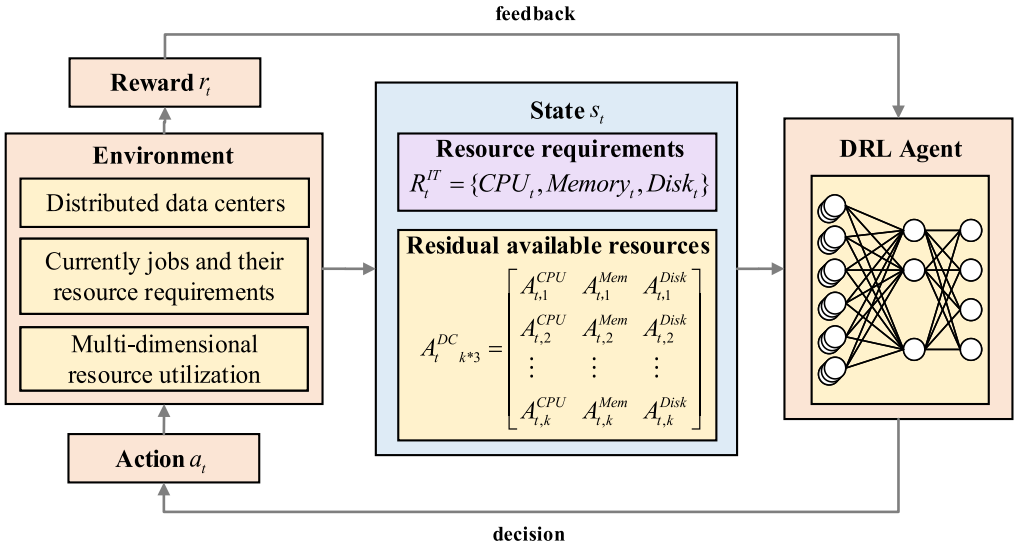


Fig. 3. Applying DRL to multi-dimensional resource allocation.

Available IT resources in distributed data centers, which are presented as:

$$\mathcal{AR}^{IT}{}_{k*3} = \begin{bmatrix} \mathcal{AR}_1^{CPU}, \mathcal{AR}_1^{Mem}, \mathcal{AR}_1^{Disk} \\ \mathcal{AR}_2^{CPU}, \mathcal{AR}_2^{Mem}, \mathcal{AR}_2^{Disk} \\ \vdots \\ \mathcal{AR}_n^{CPU}, \mathcal{AR}_n^{Mem}, \mathcal{AR}_n^{Disk} \end{bmatrix}$$

where n is the number of DCs.

2) Action Space: The action space (marked as $\mathbb{A}$) is the set of allocation decision where DRL agent must choose a certain action from it to optimize the reward function at each time unit. In this design, the goal is to optimize the decision-making of workload placement for resource allocation according to the optimization objectives in the mathematical model. Therefore, $\mathbb{A}$ is composed of the set of date centers $\mathbb{D}$. For a current job, the agent selects a candidate DC from the $\mathbb{A}$ to meet workload requirements of three dimensions of resource. The dimension of the action space is $|\mathbb{D}|$.

3) Reward Design: The agent learns how to select the specific action in the specific environment to optimize the long-term reward. The DRL-based multi-dimensional resource allocation aims to place as many as possible workloads for a given amount of resources. To achieve this goal, the rewards are designed to maximize the minimum available multi-dimension resource and maximize the balance usage of the multidimensional resources within a single data center. Therefore, three dimensions of IT resource space over distributed data centers can be fully used and a further workload can be placed and accommodated.

Specifically, the reward function at each timestep t is designed as follows:

$$\text{Reward}_t = \sum\nolimits_{D^l \in \mathbb{D}} (\mathcal{MA}_l^t + \mathcal{MB}_l^t) \tag{4}$$

where $\mathcal{MA}_t^l$ and $\mathcal{MB}_t^l$ are its minimum available multi-dimension resource and *MRBI* for a given DC $D^l \in \mathbb{D}$, respectively. The agent of NESRL-MRM only receives a value of the reward function at one timestep. Without loss of generality, we regard $\mathcal{MA}$ and $\mathcal{MB}$ as equally for simplicity in the following context.

B. Training Design With a Natural Evolution Strategy

To develop such an artificial DRL agent of multi-dimensional resource manager, a natural evolution strategy (NES) is designed to train deep neural networks to approximate the optimal action-value function in the NESRL-MRM model. This training design is inspired by recent well-known work from OpenAI [33], which applied a developed NES (hereafter referred to as OpenAI-ES) to solve standard RL benchmark problems.

NES is a class of black-box optimization algorithms, whose main idea is its heuristic search procedures inspired by natural evolution [34]. Specially, NES iteratively updates a parameterized search distribution by an estimated gradient (the natural gradient) on its parameters towards higher expected fitness. As an alternative approach to solve RL problems, NES can reliably train neural network policies and be competitive with competing RL algorithms on the hardest environments, which is well suited to be scaled up to many parallel workers in modern distributed computer systems [33].

For the proposed NESRL-MRM, training the agent for a DRL task requires multiple trials or rollouts. Instead of other DRL techniques (e.g., Q-learning based methods such as DQN and policy gradient methods such as A3C), we employ NES strategy to train the agent NESRL-MRM by population-based evolutionary search. One motivation is its competitiveness with much faster wall-clock time by distributed computation due to better parallelization of NES [35]. The other motivation is its potential advantage over diverse exploration of the population-based evolutionary search, so as to avoid falling into local optimal spaces. Moreover, the redundancy inherent in a population also facilitates robustness and stable convergence properties [36], especially when incorporated with elitism.

By interacting with the environment through a sequence of observations, the agent of NESRL-MRM is to select actions in a fashion that maximizes cumulative future reward. In the training process, NES does not calculate gradients analytically, but instead approximates the gradient of the reward function in the parameter space, similar to a finite-difference approximation of the gradient [35], [36]. With the aid of NES, NESRL-MRM's agent is capable of learning to handle such a challenging DRL-based multi-dimension management task.

We describe the training algorithm in Algorithm 1 and Algorithm 2, and its main notations are listed in Table I. The evolution strategy used in this paper is a class of black-box optimization algorithms, which is optimized repeatedly through parallel iterations [33], [35]. The parallelized nature of this training design is unique in making use of shared random seeds, which drastically reduces the bandwidth required for communication between workers. Workers refer to objects in a computing system or algorithm that can perform specific tasks in parallel, which are widely used in the parallel computing fields [37]. In this paper, worker refers to the parallel working thread established by the evolutionary strategy runtime, which is employed to execute each individual of populations in a parallel manner to optimize the algorithm performance.

In Algorithm 1, the episode is a brief unit in the training process of agent, which ends with a terminal state in a special time step, so that the agent-environment interaction breaks naturally into subsequences [38]. After one episode, with a reset to a standard starting state, the next episode begins independently of how the previous one ended. At every generation, a population is mutated after obtaining environment information and its objective function value is evaluated. The parameter vectors with highest function value are then used to form the population for the next generation. In this way, training is iterated until the objective is fully optimized, so that NES directly searches in the parameter space of neural networks to find an effective policy for this RL task.

The main idea of training design is to iteratively update parameters of DNNs using the sampled gradient of expected fitness. In lines 19th and 20th in Algorithm 1, NES algorithm perturbs parameters with the Gaussian noise to improve parameters θ_k for each worker at each iteration, so that exploration is thus driven and new individuals with better rewarding return will occur.

TABLE I
NOTATIONS FOR NESRL-MRM

Notation	Meaning
$\mathbb{B}$	dataset array
$\mathbb{D}$	the set of date centers
L_t	the current number of unallocated jobs at a time slot t
α	learning rate
σ	standard deviation of noise
$\mathbb{A}$	action space
A_t^k	action set of iteration k at time slot$t$
$\mathbb{S}$	state space
s_t	the state at time slot t
a_t	the action at time slot t
r_t	the value of reward at time slot t
$\varphi(s_t)$	the eigenvector of the state s_t
γ	the discount factor in updated function
$Q(\bullet)$	state-action function
θ_k	policy parameters at episode k,
θ_k^i	policy parameters of neural network in the i^{th} worker at iteration k
ϵ_i	the Gaussian perturbation vector in the i^{th} worker
r_i^{ep}	the total rewards for the work i

V. PERFORMANCE EVALUATION

A. Experiment Setting

The agent of NESRL-MRM was trained on a desktop computer with 6 CPU cores, 256 GB RAM, 256 GB SSD in Windows 10. NESRL-MRM is implemented using the TensorFlow version 1.14.0. We used Python 3.8 and pycharm as an IDE. In our evaluation, we assumed that every DC has a CPU with 64 cores, memory of 512GB and hard-disk I/O data rate capacity of 50Gbps.

The agent uses a fully connected DNN to characterize the relationship between states and actions. In DNN construction, we use three layers of fully-connected Exponential Linear Units (ELUs) with 30, 15 and 10 neurons, respectively, to build an autoencoder. The NES-based training algorithm (i.e., Algorithm 1) is used to guide neural network learning optimization, where γ, α, σ and N were set as 0.95, 0.05, 0.05, 50, respectively.

We have evaluated the proposed solution based on real-life trace data according Amazon EC2 instances and Google Trace, and simulation results show the effectiveness of the proposed solution according to specific performance metrics.

To demonstrate the effectiveness and robustness of NESRL-MRM, evaluations are conducted on real-life trace data according Amazon EC2 instances [39] and Google Trace [40]. In this paper, we consider an online optimizing of multi-dimensional resource in distributed data centers, where workload requests arrive and depart dynamically in an online manner and resource requirements are not known beforehand. This dynamic nature resembles a realistic scenario. We extract CPU cores, Memory and Disk requests of jobs from the traces. In the training process, the number of iterations of the models is 10,000. To provide reasonable generalization, each model was trained and evaluated against independent datasets that were extracted from both Amazon EC2 instances and Google Trace.

Amazon EC2: We consider resource requirements of workload to be equal to three different types of instances from general purpose applications provided by Amazon EC2. The resources characteristics are downloaded from the Amazon websites [39]. The instance types provided by Amazon EC2 consists of different combinations of CPU, memory, and I/O data rate capacity, which provides users with flexibility to choose the appropriate combination of resources for different applications. In our simulation, we chose 9 types of instances

Algorithm 1: Training Deep Neural Networks With Neuro-Evolution Strategy for NESRL-MRM

Input:
$\mathbb{B}$: dataset for tasks;
//It contains states of DCs and task requirements
α: learning rate;
σ: noise standard deviation;
Initialize:
n: the number of parallel workers with known random seeds;
//Construct a population with n individuals;
Q: the reward model of state-action;
θ_0: the policy parameters of the reward model Q;
Output:
The neural network parameters with the highest total reward among n parallel workers;
for *episode* $k = 1$ *to* M **do**
  /*Parallelized neuro-evolution strategy*/
  for *each worker* $i = 1$ *to* n **do**
    /* Reset the environment and generate the parameters of the model Q*/
    Initialize the environment, get initial state s_1 and its eigenvector $\phi_1 = \phi(s_1)$;
    Initialize the total rewards $r_i^{ep} = 0$;
    Sample $\epsilon_i = N(0, I)$;
    Update the parameters of neural network in the i^{th} worker $\theta_k^i = \theta_k + \sigma\epsilon_i$;
    //Add noise to neural networks;
    Use Algorithm 2 to calculate rewards by interacting with the environment;
  end
  /* Update the network parameters proportionally according to the fitnesses */
  Send all scalar returns r_i^{ep} from each worker to every other worker;
  for *each worker* $i = 1$ *to* n **do**
    Reconstruct all perturbations ϵ_j for $j = 1$ *to* n;
    Set $\theta_{t+1} = \theta_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} r_i^{ep} \epsilon_j$;
  end
end

Algorithm 2: Total Reward Calculation Involved Interaction With the Environment

Parameters:
θ_k^i : the policy parameters of the reward model Q of the i^{th} parallel worker of the k^{th} episode;

Output:
The total reward values r_i^{ep} received by each parallel workers;

for $t = 1$ *to* T **do**
  Traverse the workload dataset and calculate the number of operations to be allocated $L_t = |\mathbb{B}_t|$;
  // Obtain the current status of unallocated jobs
  for $c = 1$ *to* L_t **do**
    Calculate action values $Q\left(\varphi\left(s_t^k\right); \theta_k^i\right)$ in the action set A_t^k;
    Sort the action set A_t^k in descending order of action values $Q\left(\varphi\left(s_t^k\right); \theta_k^i\right)$;
    Read the first action in A_t^k and mark it as a_t^k;
    Let $m = a_t^k$ and allocate resources of the m^{th} data center to the job;
    Obtain the next state s_t^{k+1} and calculate reward value r_t^k;
    Update the eigenvector $\varphi_{k+1} = \varphi\left(s_t^{k+1}\right)$ by preprocessing;
    Calculate the total reward $r_i^{ep} = r_i^{ep} + r_t^k$;
    //Calculate the total reward as individual fitness;
  end
end

belong to 3 typical classes in Amazon EC2 instances for both training and testing, including the balanced instance, the computation-optimized instance and the memory-optimized instance. All types of instances with their specific requirements are generated randomly with equal probability. The arrival process of Amazon EC2 instances follows the Poisson distribution and their lifetime is subject to the negative exponential distribution. Resource allocation is implemented in accordance with the first-come-first-served (FCFS) principle. The specific requirements of EC2 instances are listed in Table II.

TABLE II
THE PROFILE OF AMAZON EC2 INSTANCES

	Instance Type	CPU (Cores)	Memory (GiB)	Disk bandwidth (Gbps)
General instance	m5.large	2	8	3.5
	m5.2xlarge	8	32	3.5
	m5.8xlarge	32	128	5
Memory optimal type	r5.large	2	16	3.5
	r5.2xlarge	8	64	3.5
	r5.4xlarge	16	128	3.5
Computing optimal type	c5.large	2	4	3.5
	c5.2xlarge	8	16	3.5
	c5.9xlarge	36	72	7

Google Trace: We use real data center workload traces from Google cluster-usage traces [40], [41] to emulate the multi-dimension resource demands of traffic, similar to the datasets in [24] and [18]. Google trace records the resource usage data of different workload in a Google data center over a month-long period, which contains the CPU, memory and disk requirements (normalized by the resource of one server) from workload, as well as arrival time (absolute time value) and durations. To simulate the workload in distributed DCs, we read open-sourced Google Trace file and split the traces into 200 segments, and each segment contains about 100,000 jobs, corresponding to the workload for a M-machine cluster. For the training, we extract CPU cores, Memory and Disk requests of jobs from the traces, and simulate the workload on a M-DC cluster by changing the number of jobs to obtain the different

workloads. We randomly selected jobs with specific requirements from the dataset file according to the load intensity based on the business requirement files.

B. Baseline Schemes

We evaluate the proposed NESRL-MRM and compare its performance against four baseline schemes, including a DRL-based resource allocation scheme according to *A2CScheduler* in [29], a threshold-based algorithm *Min-DIFF* in [18], a tight packing heuristic algorithm (termed as *TPA*), and a traditional round-robin resource allocation policy (denoted *RRA*).

A2CScheduler presented in [29] is an A2C deep reinforcement learning algorithm to address the customized job scheduling problem in data centers, which applies a reward function related to averaged job waiting time. The DRL framework of A2CScheduler consists of an actor network, a critic network and the cluster environment. Its main contribution is that DRL training process is designed to reduce the gradient estimation variance and to update parameters efficiently. Due to differences between A2CScheduler and our proposed NESRL-MRM in their application scenarios and optimization objectives, we slightly modified the reward design and datasets of A2CScheduler in terms of multiple dimension resources for the sake of fairness. The modified A2CScheduler is hereafter referred to as A2C-based Multi-dimensional Resource Management (*A2C-MRM*).

As one of the most related virtual machines placement problem in cloud DCs, a threshold-based algorithm, termed as Min-DIFF [18], is used as a baseline scheme. The main idea of Min-DIFF is to balance the usage of resources along different dimensions according to a well-designed threshold, so to reduce the risk of PM overloading in a threshold-based placement strategy. In the simulation, similarly, we set the warning line is 80% along each resource dimension, and then the threshold of each resource dimension is calculated according to the definition of threshold in [18], which depends on the warning line, the largest VM requirement threshold and total resource along dimension d of the PM.

Tetris [17] is a heuristic by the alignment of a task relative to a machine across multiple dimensions, where the alignment was calculated as the dot product between task requirements and available resources across multiple dimensions.

To demonstrate the benefit of our scheme in terms of guaranteeing the accommodation of more workload in the future, we simulate another heuristic scheme–Tight Packing Algorithm (termed as TPA)– with the aim to minimize the number of DCs to accommodate the given workload. In TPA, new workload will be placed in the current DC if there are enough resources; otherwise, the next DC with available resources will be considered as a candidate.

C. Simulation Results

In this part, we firstly evaluate the proposed *NESRL-MRM* in terms of convergence performance of training against the DRL-based baseline scheme *A2C-MRM*, and time efficiency against all baseline schemes. To demonstrate the effectiveness and robustness of NESRL-MRM, we simulate 20 tests for different traffic load from 50 to 350 using both Amazon EC2 instances and Google Trace. We compare all algorithms in terms of the minimum utilization, the multi-dimensional resource balance index (MRBI) as mentioned earlier and blocking probability, and report results on average and the 95% confidence interval.

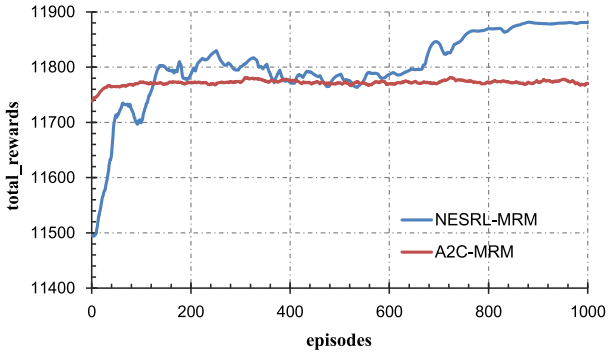


Fig. 4. Cumulative rewards of NESRL-MRM and A2C-MRM.

1) Convergence Performance of Training: We first assess convergence performance of training for both A2C-MRM and NESRL-MRM in 15 distributed DCs scenario. Training DNNs involves evaluating the neural network in iterative update manner, where the number of iterations is set 10,000 per input sequence sample. We refer to one execution of this process as a training epoch. To provide reasonable generalization, each model was trained and evaluated against independent datasets that were extracted from both Amazon EC2 instances and Google Trace.

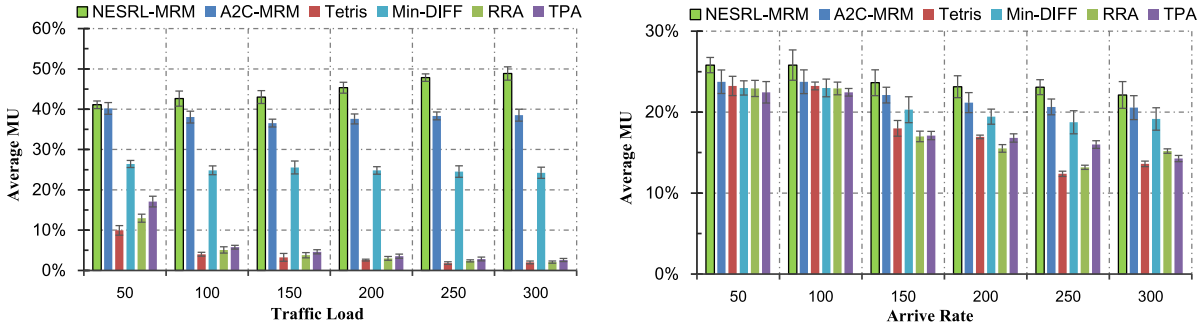
Fig. 4 show the evolutions of cumulative rewards during training. We can obtain the an observation that cumulative reward curves of both algorithms gradually tend to convergence with increased number of training epoch. Moreover, after convergence by sufficient number of epochs, NESRL-MRM can achieve slightly higher cumulative rewards than A2C-MRM in the same dateset. Our results suggest that A2C leads to ineffective learning although it is faster than, i.e., even after more than 600 learning epochs, the reward value of the produced resource allocation strategy is still lower than NESRL-MRM. It is because NESRL-MRM' potential advantage over diverse exploration of the population-based evolutionary search, so as to avoid falling into local optimal spaces.

2) Evaluation for Time Efficiency: Time efficiency is evaluated by training and reasoning decision time (testing time for DRL-based algorithm) through experimental simulations. Node processes of decision-making of each baseline scheme is executed on the same datasets. We report the training time of A2C and NESRL for DRL models in Table III, including the convergence time and the average time of each epoch. For the decision process, A2C-MRM and NESRL-MRM with their trained model, as well as non DRL-based algorithms, are evaluated according to their corresponding decision time.

As shown in Table III, NESRL-MRM needs to spend around twice more time to train its convergent DRL model than A2C-MRM, and also takes a slight more time to run

TABLE III
EVALUATION OF TIME EFFICIENCY

	Agorithm	Convergence Time	Average Time of Epoch	Decision Time
EC2 datasets	NESRL-MRM	9833.168s	11.568s	1.342ms
	A2C-MRM	3172.748s	9.614s	9.074ms
	RRA	-	-	1.064ms
	TPA	-	-	1.131ms
	Min-DIFF	-	-	1.326ms
	Tetris	-	-	1.014ms
Google Trace datasets	NESRL-MRM	30456.578s	17.915s	1.053ms
	A2C-MRM	1640.321 s	13.556s	9.36ms
	RRA	-	-	0.937ms
	TPA	-	-	0.916ms
	Min-DIFF	-	-	1.095ms
	Tetris	-	-	1.287ms



(a) EC2 instants (b) Google Trace

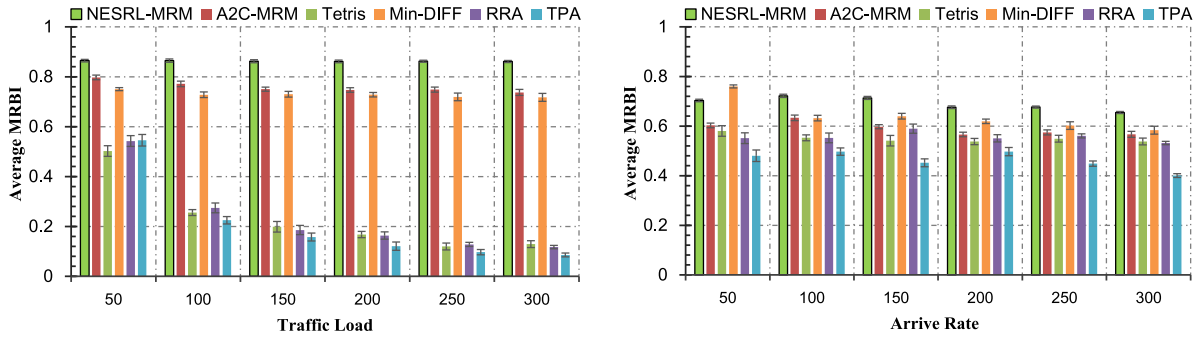
Fig. 5. Performance Comparison on average minimum utilization using (a) Amazon EC2 instants (b) Google Trace.

each epoch. The learning time is slower than the baseline solution due to diverse exploration of the population-based natural evolutionary search in the training design of NESRL-MRM. In the decision process, however, NESRL-MRM is competitive its online decision over A2C-MRM (only 15% or less of that in A2C-MRM), and its average decision time is near to those in heuristics (i.e., RRA, TPA, Min-DIFF and Tetris), which makes the proposed scheme practical in reality. Compared with A2C-MRM, the reason for the faster online decision is that NESRL-MRM has competitiveness with much faster wall-clock time by distributed computation due to better parallelization of NES. It is noteworthy that NESRL-MRM sacrifices offline training time to outperform all baseline schemes in terms of decision time on average and performances.

3) Performance Analysis in Terms of MU and MRBI: After training our model, in this part, we compare NESRL-MRM with A2C-MRM, Min-DIFF, Tetris, RRA, and TPA in terms of the minimum utilization (MU) and multi-dimension resource balance index (MRBI) using Amazon EC2 instances. All simulation were repeated 20 times and the number of distributed DCs is set as 15, which realistically resembles modern distributed datacenters cluster, such as FaceBook datacenters [42], Baidu datacenters [43].

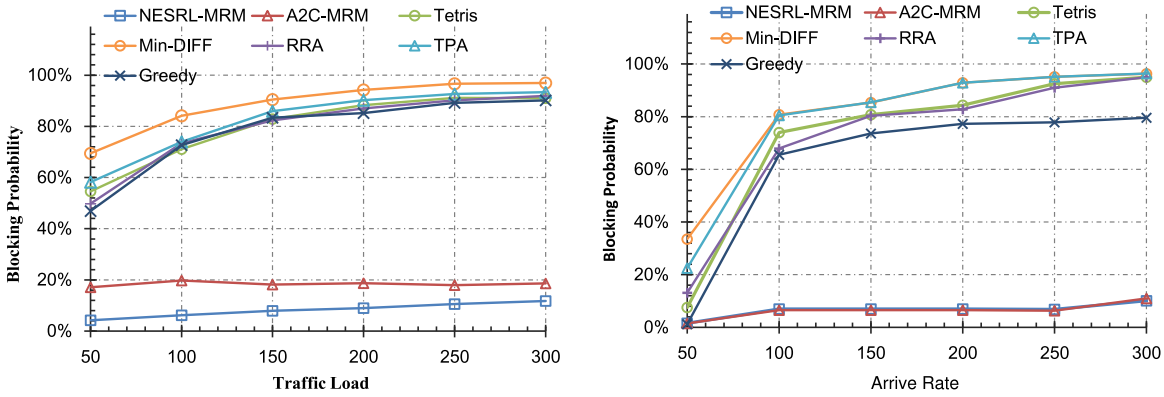
Fig. 5 shows the average of the minimum utilization (MU) along multi-dimensional resource across distributed DCs. Each bar shows the average of the minimum utilization over the 20 test instances, and error bars depict the 95% confidence interval. NESRL-MRM is observed to achieve much higher MU than those of RRA, TPA, Min-DIFF and Tetris. It is because that maximizing the minimum utilization is one pursuit of NESRL-MRM, and the agent can learn to optimize the allocation by reward feedback. Heuristics such as Tetris, min-diff, RRA and TPA just follow certain rules (such as scoring and thresholds) without considering historical experience and environment information, which makes it difficult to adapt to complex across DCs environments. But compared with A2C-MRM, it is observed that the average MU of NESRL-MRM is slightly larger with 20% performance improvement, furthermore, the performance gap is widened as the traffic load increases. The main reason is that the agent of A2C-MEM is also to maximize resource utilization, but has a disadvantage over diverse exploration against NESRL-MRM, resulting in falling into local optimal spaces.

As another significance metric, the multi-dimension resource balance index is to assess the balancing performance along three dimensional resources across all data centers, which is shown in Fig. 6. It is observed from Fig. 6 that regardless the patterns of workload, NESRL-MRM still has a higher multi-dimension resource balance index than baseline schemes, especially for Amazon EC2 instants. With the rising of traffic loads, the advantages of NESRL-MRM further increase for Amazon EC2, and its value is about more than 4 times of Teris (also considering muti-dimensional resource utilization). For testing Google Trace, the performance advantage of NESRL-MRM is less obvious compared with baseline



(a) Amazon EC2 instants (b) Google Trace

Fig. 6. Performance Comparison on the multi-dimension resource balance index using (a) Amazon EC2 instants (b) Google Trace.



(a) Amazon EC2 instants (b) Google Trace

Fig. 7. Performance Comparison on average blocking probability (a) Amazon EC2 instants (b) Google Trace.

schemes under low traffic load. This is because resources of Google clusters are so abundant that no resource fragmentation occurs. And the performance gap between NESRL-MRM and Min-DIFF is dwindled compared with Amazon EC2 testing. Min-Diff which optimized balanced resource utilization along muti-dimensional resources also outperforms than other baselines, but has poor performance than NESRL-MRM in terms of both balanced utilization and the worst available utilization. The main reason is that the agent of NESRL-MRM is also to maximize resource utilization, while the balance usage of multi-dimension resource also is considered. As a result, the remaining resource of each dimension is sufficient, rather than being wasted to meet only the individual requirement.

4) Performance Analysis in Blocking Probability Under Variable Traffic Load: Blocking probability is defined as the ratio of the total unsuccessful requests to the total arriving requests over a period of time, which is the probability that a coming request from various applications is not immediately serviced because of inadequate resources. A lower blocking probability usually indicates a more successful allocation process that can accommodate more workload requirements. Blocking probability of future incoming workload demands is the most important metric for online resource optimization where resource demands are not known beforehand [44]. Blocking probability is only evaluated the effect of decisions during a period of time from the user perspective, while MRBI and MU are the metrics to measure immediate utilization of resources at the current moment from the provider perspective. The evaluation is derived from the fact that resource allocation algorithms should avoid service level objective violations due to resource exhaustion or over utilization [30].

To justify the effectiveness of our proposed solution, we have conducted extensive experiments to evaluate blocking probability under different traffic load in the 15-DC scenario. In this particular case, additional experiments with a simple greedy baseline algorithm are presented, which searches over the feasible space in a greedy fashion with regard to MRBI only. It chooses the DC with the highest MRBI at each step of allocation. The greedy baseline algorithm is evaluated to further benchmark the effect of balanced utilization on blocking probability and explain the efficiency of our reward design.

In Fig. 7, it can be seen that the proposed NESRL-MRM significantly outperforms other algorithms with a much lower blocking probability, specifically no higher than 20%. NESRL-MRM can achieve only 40% blocking of heuristics, while 20% of those in medium and high load for Google Trace. Compared with A2C-MRM, NESRL-MRM has lower blocking probability for Amazon EC2 instants, but it is comparable blocking performance for Google traces. The main reason is that the agent of A2C-MRM is also to maximize resource utilization under consideration of multiple dimension resources, but has a different training manner against NESRL-MRM.

The other observation is the greedy algorithm has a small advantage on blocking probability comprised to other heuristics, but it is incomparable than NESRL-MRM. It is because the greedy algorithm is to maximize MRBI at each

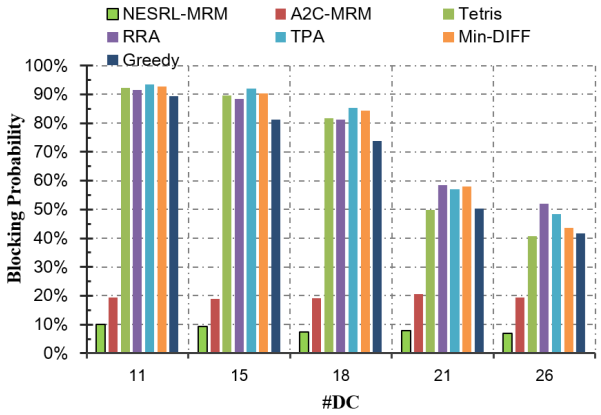


Fig. 8. Performance Comparison on average blocking probability with different numbers of DCs (11 for Apple Inc. [45], 18 for Facebook [42], 21 for Google [46] and 26 for Amazon [47]).

step of allocation, and balanced utilization along resource dimensions will bring a gain to the acceptance rate; on the other hand, the greedy algorithm only focuses balanced utilization in the most short-sighted style. While our reward design is to select the DC with the most balanced utilization along resource dimensions, as well as the more bottleneck available multi-dimensional after deploying workload.

5) Performance Analysis in Blocking Probability With Different Scale of Distributed DCs: To more comprehensively evaluate the permanence of our proposed NESRL-MRM scheme in terms of block probability, we carried out further experiment scenarios with different number of distributed DCs, ranging between 11 and 26. As shown in Fig. 8, NESRL-MRM has the lowest blocking probability (under 15%) in the testing of different numbers of DCs although its agent was trained in the 15-DC setting. Another observation is that blocking probability is decreasing with the increase in quantity of distributed DCs, this is because the total resources are set to be the same across these scenarios, then the larger number of DCs mean the less resources each DC will get. So intuitively, resource fragmentation is more likely to occur with larger number of low-capacity DCs.

VI. Conclusion

In this paper, we proposed NESRL-MRM, a DRL-based multi-dimensional resource allocation algorithm for learning the optimal online resource allocation policies in distributed data centers. Aiming at balancing multi-dimensional resource utilization and maximize available resources, NESRL-MRM parameterizes multi-dimensional resource allocation policies with DNNs based on DRL framework. To train such an DRL agent with much faster wall-clock time and diverse exploration, we developed a training mechanism and search space dynamicity based on a natural evolution strategy (NES) to train DNNs progressively with its experience from dynamic resource provisioning, rather than an accurate mathematical model. Based on Amazon EC2 and Google data traces, our simulation results show that NESRL-MRM achieves a more balanced use of resources along multiple resource dimensions, which significantly reduces blocking probability compared with heuristic baseline schemes and obtains competitive online decisions with a 85 percent improvement in time efficiency compared with A2C-MRM.

References

[1] B. Wan, J. Dang, Z. Li, H. Gong, F. Zhang, and S. Oh, "Modeling analysis and cost-performance ratio optimization of virtual machine scheduling in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1518–1532, Jul. 2020.

[2] C. Guerrero, I. Lera, B. B. Gonzalez, and C. Juiz, "Multi-objective optimization for virtual machine allocation and replica placement in virtualized hadoop," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2568–2581, Nov. 2018.

[3] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[4] D. Jiang, Y. Wang, Z. Lv, W. Wang, and H. Wang, "An energy-efficient networking approach in cloud services for IIoT networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 928–941, May 2020.

[5] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, no. 5, pp. 36–42, Sep./Oct. 2015.

[6] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y. C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020.

[7] Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1871–1879.

[8] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 270–288.

[9] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, 2020, pp. 154–171.

[10] Z. Peng, J. Lin, D. Cui, Q. Li, and J. He, "A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm," *Clust. Comput.*, vol. 23, pp. 2753–2767, Dec. 2020.

[11] J. Feng, W. Zhang, Q. Pei, J. Wu, and X. Lin, "Heterogeneous computation and resource allocation for wireless powered federated edge learning systems," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3220–3233, May 2022.

[12] Z. Hu, B. Li, and J. Luo, "Time- and cost- efficient task scheduling across geo-distributed data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 3, pp. 705–718, Mar. 2018.

[13] Q. Zhang and W. Shi, "Energy-efficient workload placement in enterprise datacenters," *Computer*, vol. 49, no. 2, pp. 46–52, Feb. 2016.

[14] D. Erickson, B. Heller, N. McKeown, and M. Rosenblum, "Using network knowledge to improve workload performance in Virtualized data Centers," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2014, pp. 185–194. [Online]. Available: https://doi.org/10.1109/IC2E.2014.81

[15] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica, "HUG: Multi-resource fairness for correlated and elastic demands," in *Proc. 13th Usenix Conf. Networked Syst. Design Implement.*, 2016, pp. 407–424.

[16] L. Chen, S. Liu, B. Li, and B. Li, "Scheduling jobs across geo-distributed datacenters with max-min fairness," *IEEE Trans. Netw. Sci. Eng.*,, vol. 6, no. 3, pp. 488–500, Jul.–Sep. 2019.

[17] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 455–466, Aug. 2014.

[18] S. Ji, M. Li, N. Ji, and B. Li, "An online virtual machine placement algorithm in an over-committed cloud," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2018, pp. 106–112.

[19] W. Wei, H. Gu, W. Deng, Z. Xiao, and X. Ren, "ABL-TC: A lightweight design for network traffic classification empowered by deep learning," *Neurocomputing*, vol. 489, pp. 333–344, Jun. 2022.

[20] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proc. 16th ACM Workshop Hot Topics Netw.*, 2017, pp. 185–191.

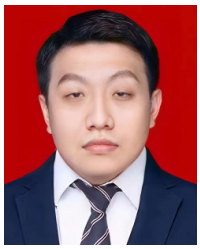
[21] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1325–1336, Jun. 2019.

[22] W. Wei, H. Gu, and B. Li, "Congestion control: A renaissance with machine learning," *IEEE Netw.*, vol. 35, no. 4, pp. 262–269, Jul./Aug. 2021.
[23] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, Oct. 2020.
[24] N. Liu et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 372–382.
[25] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Comput.*, vol. 4, no. 6, pp. 60–69, Nov./Dec. 2017.
[26] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.
[27] S. Bian, X. Huang, and Z. Shao, "Online task scheduling for fog computing with multi-resource fairness," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–5.
[28] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, Nov. 2016, pp. 50–56.
[29] S. Liang, Z. Yang, F. Jin, and Y. Chen, "Data centers job scheduling with deep reinforcement learning," in *Proc. 24th Pacific-Asia Conf. Knowl. Disc. Data Min. (PAKDD)*, May 2020, pp. 906–917.
[30] S. Mitra, S. S. Mondal, and N. Sheoran, "Scheduling of time-varying workloads using reinforcement learning," in *Proc. 35th AAAI Conf. Artif. Intell. (AAAI)*, 2021, pp. 9000–9008.
[31] J. Nirjhor et al. "Arithmetic mean—Geometric mean." Accessed: 2014. [Online]. Available: https://brilliant.org/wiki/arithmetic-mean-geometric-mean/
[32] B. Carlson, "The logarithmic mean," *Amer. Math. Monthly*, vol. 79, pp. 615–618, Jun. 1972.
[33] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.
[34] D. Wierstra et al., "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, pp. 949–980, Mar. 2014.
[35] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5032–5043.
[36] Z. Wang, C. Chen, and D. Dong, "Instance weighted incremental evolution strategies for reinforcement learning in dynamic environments," 2020, *arXiv:2010.04605*.
[37] A. G. V. Kumar, A. Grama, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms*, Benjamin-Cummings, San Francisco, CA, USA, Jan. 1994.
[38] P. Babington, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, Nov. 2018.
[39] "Amazon EC2 instants." Accessed: 2006. [Online]. Available: https://aws.amazon.com/ec2/instance-types/
[40] C. Reiss, J. Wilkes, and J. L. Hellerstein. "Google cluster-usage traces: Format + schema." Accessed: 2014. [Online]. Available: https://ode.google.com/p/googleclusterdata/wiki/TraceVersion2
[41] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, Oct. 2012, pp. 1–13.
[42] M. Zhang. "Facebook's 18 data Centers: $20bn investment, 40m square feet." 2021. [Online]. Available: https://dgtlinfra.com/facebook-18-data-centers-20bn-investment/
[43] C. Sbeglia. "Baidu to increase investment in AI and cloud, plans for 5 million servers." 2020. [Online]. Available: https://www.rcrwireless.com/20200622/business/baidu-ai-cloud-plans-5-million-servers
[44] C. K. Dehury and P. K. Sahoo, "DYVINE: Fitness-based dynamic virtual network embedding in cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1029–1045, May 2019.
[45] "iCloud." 2020. [Online]. Available: https://en.wikipedia.org/wiki/ICloud
[46] "The secret cost of Google's data centers: Billions of gallons of water to cool servers." 2021. [Online]. Available: https://time.com/5814276/google-data-centers-water/
[47] "Global infrastructure of AWS." 2021. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/?p=ngi&loc=0

Wenting Wei (Member, IEEE) received the M.E. and Ph.D. degrees in telecommunication and information systems from Xidian University in 2014 and 2019, respectively. Since 2019, she has been working with the State Key Laboratory of ISN, Xidian University. Her main research interests include data center networking, network virtualization, cloud computing, and intelligent networking.

Huaxi Gu is a Professor with the State Key Laboratory of ISN, Xidian University. He is the Leader of Youth Innovation Team, Shaanxi University. He is Leading a Project as the Principal Investigator, funded by the National Key Research and Development Program of China. He is also the Principal Investigator for one key, two general, and one youth project funded by National Natural Science Foundation. He has published over 200 journal and conference papers, with his research interests being in the areas of networking technologies, network on chip, and optical interconnect. He served as the TPC Member of GLOBECOM, ICC, and PDCAT and the Technical Reviewer for multiple journals, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE/OSA JOURNAL OF LIGHTWAVE TECHNOLOGY.

Kun Wang received the B.E. and M.E. degrees in computer science and technology from Xidian University, Xi'an, China, in 2003 and 2006, respectively, where she is currently a Lecturer with the Department of Computer Science. Her current interests include high performance computing and cloud computing and the network virtualization technology.



Jianjia Li received the M.E. degree from Xidian University in 2022. From 2019 to 2022, he has been engaged in research on data center network and intelligent networking with the Advanced Network Technology Laboratory, Xidian University.

Xuan Zhang received the B.E. degree in telecommunication engineering from Xidian University in 2020. He is currently pursuing the M.S. degree with ECE Department, University of California at Davis, Davis. From 2020 to 2021, he worked with the Advanced Networking Technology Laboratory, State Key Laboratory of ISN, Xidian University. His main research interests include reinforcement learning and intelligent networking.

Ning Wang (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the Centre for Communication Systems Research, University of Surrey, U.K., in 2004, where he is currently a Full Professor with the Institute for Communication Systems. His research interests include future network design, 5G networks, network optimizations and network, and service management.