# Deep Reinforcement Learning Based Container Cluster Placement Strategy in Edge Computing Environment

1st Zhuo Chen
*College of Computer Science and Engineering*
*Chongqing University Of Technology*
Chongqing, China
chenzhuo@cqut.edu.cn

2nd Bowen Zhu
*College of Computer Science and Engineering*
*Chongqing University Of Technology*
Chongqing, China
zbw@2020.cqut.edu.cn

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001234

***Abstract*—Container-based virtualization technology has been increasingly used in edge clouds recently due to its advantages of lighter resource occupation, faster startup capability, and better resource utilization efficiency. For a task request, it is usually necessary to interconnect multiple containers to build a Container Cluster (CC), and then deploy it to edge service nodes with relatively limited resources. However, the increasingly complex and time-varying nature of tasks brings great challenges to optimally deploying CC on edge nodes. From the perspective of edge service provider, this paper regards the charges for various resources occupied by providing services as revenue, and regards the service efficiency and energy consumption generated by providing services as cost, then formulate an integer optimization model to describe the optimal deployment of CC on distributed edge nodes. Different from the heuristic-based methods adopted in existing works, we introduce Graph Convolutional Network (GCN) to extract features from the logical topological link relationships among containers in CCs and use them as the input of the solution framework to improve the solution quality. Compared with other related algorithms, the results show that the method proposed in this paper can obtain better solution quality and maintain acceptable solution efficiency under the same edge cloud environment. The improved system revenue for edge service provider can be further obtained.**

***Index Terms*—Edge computing, Network virtualization, Cloud container cluster, Neural Combinatorial optimization, Graph convolutional network**

## I. Introduction

Edge computing solves the problems of high latency and large bandwidth occupancy in the traditional cloud computing mode by applying cloud computing to the edge of the network [1]–[3]. Typically they are built on clusters of servers, combined with internal application and system software. In the past, cloud resources were usually packaged into different types of VMs to serve cloud users. More recently, cloud containers have emerged as a lighter-weight alternative to VMs. The VM needs to simulate all the resources of a physical machine, which adds unnecessary computation to the physical machine. However, container only isolates various resources required by the application at the operating system level.



In addition to purchasing a single container for computing, users often need a set of containers and a network between them to create a CC to build a reliable and scalable distributed system. Typical examples include geographically distributed machine learning systems, and service chains in network function virtualization environments. By analogy with the placement of service function chain(SFC) in a network virtualization environment, we can turn the CC placement problem in the edge computing environment into a container cluster forward graph embedding problem(CC-FGE) [4], that is, how to map all the containers contained in the CC and the links for establishing communication between containers to the underlying physical network environment. One of the main challenges faced by this technology is to place all containers in the CC to the appropriate underlying network infrastructure without breaking resource constraints in a limited time, while meeting some specific conditions of edge computing service providers, such as maximizing revenue, minimizing energy consumption or optimizing user experience [5], [6].

A classic heuristic method for solving the function embedding problem of virtual networks is the D-ViNE and R-ViNE algorithms [7], proposed by Chowdhury et al. This algorithm describes the virtual network embedding problem as a mixed integer programming problem based on the clear separation between host mapping and link mapping phases that has been done in the past. Then, the relaxation technique is used to turn it into a linear programming problem, and D-ViNE and R-ViNE algorithms are designed using deterministic and random rounding techniques, respectively. The results show that these two algorithms improve the embedding accuracy and revenue of service provider, while reducing the cost of the underlying network.Considering the link relationship between virtual network functions(VNFs) in SFC, Li Dan et al. proposed that SFC be combined into functional topology diagrams, and the weighted graph matching between functional and physical topologies is obtained by using the adjacency matrix eigenvector decomposition algorithm, and the matching results are further

optimized by the hill-climbing algorithm. The results show that the proposed method reduces the bandwidth required by the service functional chain, optimizes the balance between host load and link bandwidth, supports more service requests, and has low complexity.

The above-mentioned methods for solving VNF embedding are all based on heuristic or meta-heuristic algorithms, which have the advantage of having lower latency. However, heuristic or meta-heuristic algorithms often fail to meet the needs of edge computing service providers and users when service requests increase or resources are highly constrained. Because these problems can be formalized as a combinatorial optimization problem, both the VNF in the SFC and the containers in the CC in this paper need to be placed on the network infrastructure that satisfies the service level agreement(SLA), which was proved to be an NP problem as early as 2017 [8]. Therefore, when the number of constraints is large and the feasible domain of the solution is small, heuristic or meta-heuristic algorithms often fails to solve such problems.

In order to solve the problem of virtual network embedding when there are numerous constraints and the feasible domain of the solution is small, researchers begin to find new way. In recent years, Bello et al. proposed using Reinforcement Learning(RL) to solve combinatorial optimization problem, namely neural combinatorial optimization [9]. Quickly, Solozabal et al. applied this theory to the virtual network embedding problem in the context of SFC and obtained advantages that heuristic or meta-heuristic algorithm do not have [4]. Inspired by Andrej et al. extracting picture features through Convolutional Neural Network(CNN) and combining Recurrent Neural Network(RNN) to achieve automatic picture description, we proposed to use Graph Convolutional Network(GCN) to extract the topological links between containers in CC, and then use NCO theory to solve the CC-FGE, named GNCO.

In a word, we innovatively propose to use GCN extract the topological characteristic of CC, and then the placement strategy of CC given by extending the NCO. The rest of the paper is organized as follows. Section II defines the multi-objective combinatorial optimization problem. We elaborate the proposed algorithm in Section III and present the simulation experimental scenario and experimental results in Section IV. Finally, we draw a conclusion in Section V.

## II. Problem Definition and Modeling Analysis

In this section, we will give a detailed description of problem mentioned above. For easy viewing, we summarize the commonly used representation symbols in Table I.

### A. Container Cluster Embedding

The edge computing platform may receive different numbers of service requests at the same time, and each service request needs to realize different functions. Also, there are uncertain communication requirements between different containers in the same CC. The most intuitive impact of the scale and type

TABLE I: Notion of Optimization

| | |
|---|---|
| $G_c$ | income for per unit computing resource |
| $G_m$ | income for per unit memory resource |
| $G_s$ | income for per unit storage resource |
| $V_i$ | set of containers for service request $i$ |
| $N$ | set of host |
| $I$ | set of request |
| $c$ | expenditure for per unit power consumption |
| $\eta_{k,c}$ | computing resource utilization of host $k$ |
| $W_k^{max}$ | power consumption of of host $k$ |
| $W_k^{min}$ | idle power consumption of host $k$ |
| $X_{j,k}^i$ | binary placement variable for container $j$ of request $i$ in host $k$ |
| $a_{j,r(c)}^i$ | computing resources requested by container $j$ of request $i$ |
| $a_{j,r(m)^i}$ | memory resources requested by container $j$ of request $i$ |
| $a_{j,r(s)^i}$ | storage resources requested by container $j$ of request $i$ |
| $\phi_{m,n}^i$ | bandwidth demanded by container $m$ and container $n$ of CC $i$ |
| $B_{k_u,k_v}$ | bandwidth between direct link of host $k_u$ and $k_v$ |
| $R_k^c$ | computing resource of host $k$ |
| $R_k^m$ | memory resource of host $k$ |
| $R_k^s$ | storage resource of host $k$ |
| $u_k$ | binary activation variable for host $k$ |

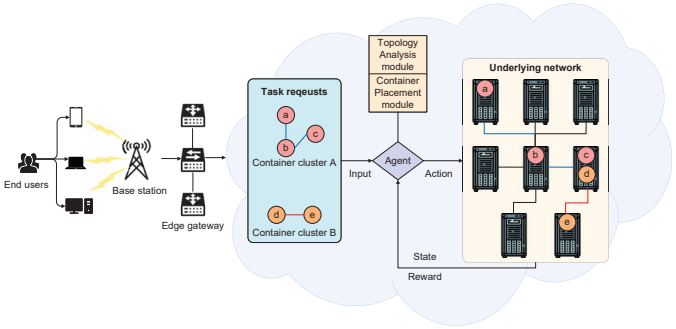



Fig. 1: Container cluster placement in edge cloud

of service requests is the change of virtual host and links. Workload fluctuations usually change the number of resource requirements of virtual host or links, that is, the change of resource configuration [10]. Fig.1 shows the process of mapping two different CC to the underlying physical network.

We assume that at the initial time $t = 0$, the system receives two service requests $A$ and $B$ from the end user, the system needs to consider the reasonable mapping of *container cluster A* and *container cluster B* corresponding to the service request to underlying physical network. When the system resources meet the service request requirements, in particular, since both the CC and the underlying physical network are in the form of graph structures, a single host that satisfies the resource is not necessarily suitable for container mapping. Because the communication requirements between containers, it is also necessary

to check whether the whole link between physical hosts meet the bandwidth requirements generated by the communication between two containers. While everything is satisfied, the system will map the container to the corresponding physical host according to the placement strategy given by the agent, and the underlying physical network state will also make corresponding changes to deal with the service request at the next time.

### B. Optimization Model and Problem Description

From the perspective of edge computing service providers, we build an optimization model, hoping to reduce the total energy consumption expenditure on the premise of meeting user service requests as much as possible, so as to maximize the profit of service providers:

$$max(profit - expenditure) \tag{1}$$

In Eq.2, the IT resources occupied by the container $i \in I$ for providing services are respectively represented as follows: computing resource $a^C_{i,j}$, memory resource $a^M_{i,j}$, and storage resource $a^S_{i,j}$. In addition, considering different charging modes, the charging coefficients of three resources are defined respectively: $G_c$, $G_m$ and $G_s$. To balance the relationship between energy consumption and user experience, we introduce the service effect coefficient $(1 - \eta_{k,c})$. [11], [12].

$$\begin{aligned} profit = \sum_{k\in N}\Big[G_c(1-\eta_{k,c})\cdot\sum_{i\in I}\sum_{j\in V_i}X^i_{j,k})\cdot a^i_{j,r(c)} \\ + G_m\cdot\sum_{i\in I}\sum_{j\in V_i}X^i_{j,k}\cdot a^i_{j,r(m)} + G_s\cdot\sum_{i\in I}\sum_{j\in V_i}X^i_{j,k}\cdot a^i_{j,r(s)}\Big] \end{aligned} \tag{2}$$

In Eq.3, $W^{max}_k$ and $W^{min}_k$ is defined to represent the maximum and minimum energy consumption of edge service node $k$, respectively. Since energy consumption is positively related to resource usage, we use $(W^{max}_k - W^{min}_k)\cdot\eta_{k,c}$ to denote the energy consumption of the edge service node $k$ in the working status. At the same time, the energy consumption $W^{min}_k \cdot u_k$ when the physical node $k$ is idle is also considered in (3). In addition, we define $C$ as the energy consumption coefficient to obtain the actual expenditure of edge service provider.

$$expenditure = \sum_{k\in N}\Big[(W^{max}_k - W^{min}_k)\cdot\eta_{k,c} + W^{min}_k\cdot u_k\Big]\cdot C \tag{3}$$

The optimization model is limited by several constraints. Eq.4 represents the utilization $\eta_{k,c}$ of computing resources on physical host $k$, and the range of value is limited to $[0, 1]$.

$$\eta_{k,c} = \frac{\sum\limits_{i\in I}\sum\limits_{j\in V_i}X^i_{j,k}\cdot a^i_{j,r(c)}}{R^c_k} \tag{4}$$

Eq.5 restricts that the container $j$ of the service request $i$ can only be placed on one physical host and cannot be placed repeatedly.

$$\sum_{k\in N}X^i_{j,k} \le 1, \qquad \forall\ i\in I,\ j\in V_i,\ K\in N \tag{5}$$

Eq.6 limits that the bandwidth resources occupied by the communication between the two containers $m$ and $m$ of the service request $i$, which are located between the physical host $k_u$ and $k_v$, respectively, do not exceed the total bandwidth resources between the physical host $k_u$ and $k_v$.

$$\sum_{i\in I}\sum_{m,n\in V_i}\phi^i_{m,n}\cdot X^i_{m,k_u}\cdot X^i_{n,k_v} \le B_{k_u,k_v} \tag{6}$$

Eq.7, Eq.8 and Eq.9 respectively limit that the sum of all container resources contained in the service request do not exceed the total amount of computing resources, memory resources and storage resources.

$$\sum_{i\in I}\sum_{k\in N}X^i_{j,k}\cdot a^i_{j,i(c)} \le R^c_k \tag{7}$$

$$\sum_{i\in I}\sum_{k\in N}X^i_{j,k}\cdot a^i_{j,i(m)} \le R^m_k \tag{8}$$

$$\sum_{i\in I}\sum_{k\in N}X^i_{j,k}\cdot a^i_{j,i(s)} \le R^s_k \tag{9}$$

## III. Reinforcement Learning Solving Framework Combined with Graph Convolutional Networks

In this work, based on the NCO theory, we extend GCN to extract the topological link relationship existing in the CC, so that the agent can learn the topology structure of the CC in advance, thus giving more accurate placement strategies. Specifically, we use GCN [13] and sequence to sequence(Seq2Seq) model based on encoder-decoder structure to infer placement strategies.

### A. The Description of Solving Framework

We show the reinforcement learning decision-reward feedback loop for our problem in Fig.2. The GCN extracts the topology in the CC, and the Sequence to Sequence model consists of an encoder-decoder and a Bahdanau attention mechanism. When a given CC is input into the decision system, the agent will make appropriate decisions $A_t$ based on the network state $S_t$ dictates where the containers in the CC should be placed. The environment then estimates the placement strategy and return a feedback(rewards) $R_{t+1}$ indicating the quality of the placement strategy, and the environment updates the new State $S_{t+1}$ after placed. Baseline will give the profit expectation according to the current input.

### B. Topological Relation Description Based on Graph Convolutional Network

Suppose a CC is represented by a graph $G = (N, E)$. Where $N$ represents the node in the graph, that is, the container in the CC, and $E$ represents the edges in the graph, that is, the links generated by the communication between the containers in the CC. Then the features of nodes in $G$ form an $N \times D$ matrix

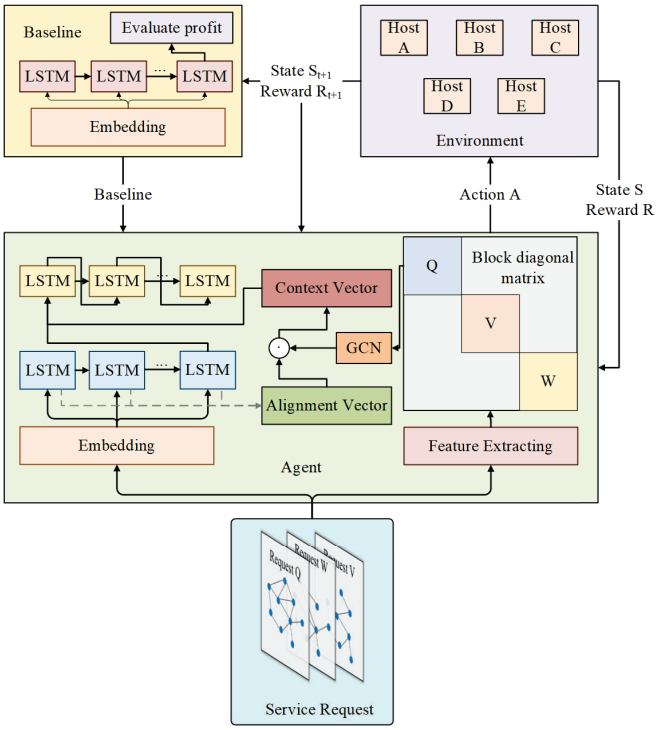



Fig. 2: Network model and reinforcement learning decision-reward feedback loop

**Algorithm 1** Description of training process of container cluster placement GNCO

**Require:** Container cluster $C$, training epoch $epoch$, learning rate of *agent* $l_a$, learning rate of *baseline* $l_c$

Initialize environment State $S_t$ , agent parameter $\theta$, baseline parameter $\sigma$
**for all** $ep \in (1, 2, ...epoch)$ **do**
  $c_i \leftarrow SampleInput(C)$ $where\ i \in (1, 2, ..., B)$
  $p_i \leftarrow SampleSolution(\pi_\theta(\cdot|c))$ $where\ i \in (1, 2, ..., B)$
  Lagrangian: $L(p_i) \leftarrow environment(p_i)$ $where\ i \in (1, 2, ..., B)$
  Baseline: $b_\sigma(c_i) \leftarrow baseline(c_i)$ $where\ i \in (1, 2, ..., B)$
  Lagrangian Approximate Gradient: $\nabla_\theta J_L^\pi(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}(L(p_i|c_i) - b_\sigma(c_i)) \cdot \nabla_\theta log\pi_\theta(p_i|c_i)$ //Monte Carlo sampling
  Calculate the loss function $\mathcal{L}(\sigma) = \frac{1}{B}\sum_{i=1}^{B}(L(p_i|c_i) - b_\sigma)^2$ //Mean squared error
  $\theta \leftarrow Adam\ (\theta, \nabla_\theta J_L^\pi(\theta))$
  $\sigma \leftarrow Adam\ (\sigma, \mathcal{L}(\sigma))$
  $\theta_{k+1} = \theta_k + \alpha \cdot \nabla_\theta J_L^\pi(\theta)$ //update agent parameter $\theta$
  $\sigma_{k+1} = \sigma_k + \alpha \cdot \mathcal{L}(\sigma)$ //update baseline parameter $\sigma$
**end for**
**return** $\theta_{k+1}, \sigma_{k+1}$

$X$, where $D$ represents the number of features. The relationship between the containers is represented by an $N \times N$ dimensional matrix $A$, which is the adjacency matrix of $G$. The hierarchical propagation of GCN is shown in Eq.10 [14].

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{10}$$

In Eq.10, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $G$ with additional self-connections. $\tilde{D} = \sum_i \tilde{A}_{ij}$ is the degree matrix of matrix $A$. $W^{(l)}$ is the training parameter matrix for the $l$ layer. $\sigma$ represents the activation function, such as ReLU, Sigmoid, etc.(in our model we use ReLU). $H^{(l)}$ is the feature of $l$ layer, for the input $H = X$.

### C. Constrained Optimization Based on Policy Gradients

Assuming that the set of CC is represented by $C$, then for a given CC $c \in C$ the policy function could be expressed as $\pi(p|c)$, which gives a higher probability to high profit placement strategy $p$ and a lower probability to low profit placement strategy $p$. In Eq.11, we use the chain rule to decompose the policy function in the neural network:

$$\pi_\theta(p|c) = \prod_{f=1}^{m} \pi_\theta(p_f|p_{(<f)}, c) \tag{11}$$

In Eq.11, $\theta$ is the neural network parameter that needs to be trained, and the probability of the placement strategy $p_f$ for the input container $f$ depends on the previous container placement position $p_{(<f)}$ and the system state. The output of the policy function is only the probability of indicating the placement location of the container, and then an objective function is defined to describe the quality of the strategy.

$$J_P^\pi(\theta|c) = \mathop{E}_{p\sim\pi_\theta(\cdot|c)}[E(p)] \tag{12}$$

The agent infers the placement strategy based on all CCs, so the profit expectation can be defined as the expectation of the container probability distribution:

$$J_P^\pi(\theta) = \mathop{E}_{c\sim C}[J_P^\pi(\theta|c)] \tag{13}$$

Similarly, the expected punishment for breaking constraints can be expressed as shown in Eq.14:

$$J_{P'}^\pi(\theta) = \mathop{E}_{c\sim C}[J_{P'}^\pi(\theta|c)] \tag{14}$$

Next, use Lagrangian relaxation techniques to transform the target problem into an unconstrained problem, in which the infeasible solution given by the agent will be punished accordingly:

$$\begin{aligned} g(\lambda) &= \max_\theta J_L\pi(\lambda, \theta) = \max_\theta \left[J_P^\pi(\theta) + \sum_i \lambda_i \cdot J_{P_i'}^\pi(\theta)\right] \\ &= \max_\theta \left[J_P^\pi(\theta) + J_\xi^\pi(\theta)\right] \end{aligned} \tag{15}$$

Where $J_L\pi(\lambda,\theta)$ is the Lagrangian objective function, $J_{P'}^{\pi}$ is the dissatisfaction signals, $g(\lambda)$ is the Lagrangian dual function, $\lambda_i$ is the Lagrangian multiplier, in other words, the penalty coefficient of the three constraint dissatisfaction signals, and $J_{\xi}^{\pi}(\theta)$ is the weighted sum of all expectations of the three constraint dissatisfaction signals. In our work, we manually set the Lagrangian multipliers $\lambda_i$.

The Lagrangian gradient $\nabla_\theta J_L^{\pi}(\theta)$ is derived by log-likelihood method with reference to the work of Williams et al. [15]:

$$\nabla_\theta J_L^{\pi}(\theta) = \underset{p\sim\pi_\theta(\cdot|c)}{E}[L(p|c)\cdot\nabla_\theta log\pi_\theta(p|c)] \tag{16}$$

Where $L(p|C)$ represents the Lagrangian value, calculated by adding the weighted sum of all constraint dissatisfaction values $C(p|C)$ to the profit value $E(p|C)$. Then, Monte Carlo sampling is used to approximate the Lagrangian gradient $\nabla_\theta J_L^{\pi}(\theta)$, where $B$ is the number of samples. In order to reduce the variance of the gradient and speed up the convergence of the model, we introduce a baseline evaluator $b_\sigma$, which composed of a simple RNN network. Finally, the Lagrangian gradient is expressed as shown in Eq.17.

$$\nabla_\theta J_L^{\pi}(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}(L(p_i|c_i) - b_\sigma(c_i))\cdot\nabla_\theta log\pi_\theta(p_i|c_i) \tag{17}$$

The baseline $b_\sigma$ estimates the profit $b_\sigma(c)$ of the current CC, and then the parameter $\sigma$ of the baseline is updated using stochastic gradient descent based on the mean square error(MSE) of $b_\sigma(c)$ and Lagrangian value $L(p|c)$ obtained in the environment.

$$\mathcal{L}(\sigma) = \frac{1}{B}\sum_{i=1}^{B}(L(p_i|c_i) - b_\sigma)^2 \tag{18}$$

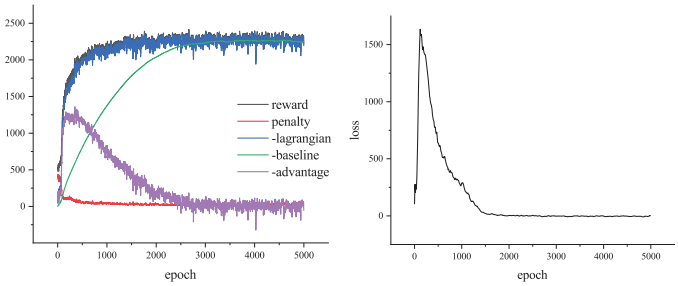
The proposed container cluster placement algorithm training process is shown in Algorithm 1.

## IV. Performance Evaluation and Analysis

In this section, we verify that the CC placement GNCO can reasonably extract topological features of CCs and make container placement decisions through a large number of simulation experiments and comparisons with other placement algorithms.

### A. Experimental Setup

In order to evaluate the performance of the GNCO, we compare it in detail with two heuristic algorithm and a simple NCO algorithm in different scale environments. The two heuristic algorithms are First-Fit Scheduling Algorithm(FFSA) [16], [17] and User-First Scheduling Algorithm(UFSA) [18]. FFSA circularly traverses the hosts and place the container on the host that meets the cloud resources and the communication requirements with the previously placed container for the first time. From the perspective of user experience, UFSA circularly traverses the hosts with low to high computing resource occupancy, and place containers on host that meet cloud resources and communication requirements with the previously placed containers for the first time. We call the number of service requests at the same time as the service scale, and the number of containers contained in a CC as the container cluster scale. For simplicity, we use symbols to represent the scale of service requests. For instance, $[1-2][5-10]$ means that the number of service requests at the same time is limited to the range of $[1-2]$, and the number of containers contained in the CC is in the range of $[5-10]$, the specific number is randomly generated.




(a) [1-2][5-10]Learning history (b) [1-2][5-10]Loss

Fig. 3: Learning history on first experimental instance

### B. Model Convergence

We tested the convergence of the GNCO in the above four experimental instances. Due to length limitation, here we give representative experimental result, which are shown in Fig.3. It should be noted that Lagrangian multipliers are positive numbers, therefore, the Lagrangian objective function value *lagrangian*, penalty value *penalty* and the difference value *advantage* between the *lagrangian* and the baseline for each iteration are often negative numbers. For convenience of viewing, In the Figure, we reverse the above values.

At the beginning of training, the agent will generate some strategies with high punishment. With the progress of learning, the agent will update the weight through stochastic gradient descent, and continuously improve the generation strategy. At the end of training, the penalty value has been greatly reduced. In the small problem instance, i.e., first set of experimental instance, as the training ends, the penalty value approaches zero, and the Lagrangian value approaches the reward value. In practice, this phenomenon is that due to sufficient resources, the system can meet most of the service requests.

### C. Placement Error Rate

Next, we compared the placement error rate of the four algorithms and test 128 experiments in under the four experimental instances to balance the contingency. When the placement strategy given by the algorithm break the constraints and the CC is not correctly placed, it is recorded as a placement error. The ratio of the number of CCs with all placement errors in the 128 experiments to the total number of CCs in 128 experiments is called the placement error rate. The experimental results are shown in Fig.4.

With the increase of service scale and container cluster scale, the problem of insufficient resources is exposed in the placement error rate. Because UFSA is user experience-oriented, the placement error rate of UFSA is the best among the four scheduling algorithms. It can be seen that except in the second set of experimental instance, the placement error rate of GNCO is lower than that of the NCO algorithm. The reason for this if that some service requests with lower profit in second set of experimental instance are selectively discarded by the GNCO algorithm, so that the resources can be preferentially allocated to the service requests with higher profit when the resources are relatively fixed.

### D. Placement profits

At mentioned at the beginning, our optimization goal is to maximize the profit of the cloud service provider. We compare the cumulative profit of the four algorithms with 128 groups of randomly generated request in four sets of experimental instance. Due to length limitation, here we only show representative experimental results(fourth experimental instance), as shown in Fig.5.

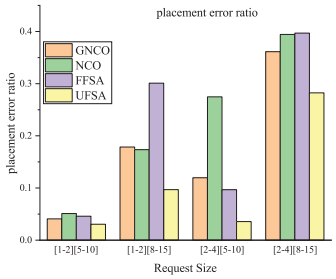



Fig. 4: placement error rate comparison of four algorithms

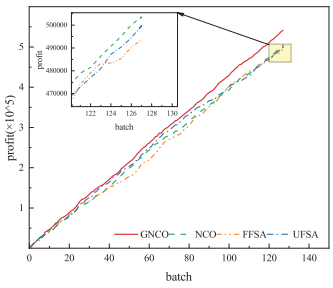



Fig. 5: Cumulative profit on fourth instance

GNCO can place the container that need communication on the same physical host as much as possible because it knows the topology in advance, so as to reduce the occupation of physical link bandwidth resources, so as to place more CC under the same physical resources and meet the service level agreement, so as to improve the overall profit of service providers.

## V. Conclusions

In this work, based on the fact that CCs and underlying physical network both are topology structures, with the goal of maximizing the cumulative profit of cloud service providers, we propose to use GCN to extract topological features of CCs, combined with reinforcement learning to solve the CC-FGE problem. For that purpose, based on the work of Solozabal et al., we extend the NCO theory and modeled the proposed problem. After testing in different experimental instances, the results show that the proposed GNCO algorithm can extract the topological features of CCs, and has obvious advantages when the container cluster scale is large.

## References

[1] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[2] Z. Chen and Z. Zhou, "Dynamic task caching and computation offloading for mobile edge computing," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[3] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Joint data offloading and resource allocation for multi-cloud heterogeneous mobile edge computing using multi-agent reinforcement learning," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[4] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2020.

[5] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[6] M. Dolati, S. B. Hassanpour, M. Ghaderi, and A. Khonsari, "Deepvine: Virtual network embedding with deep reinforcement learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 879–885.

[7] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[8] A. Marotta, F. D'andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5g virtualized infrastructures," *Computer Networks*, vol. 125, pp. 64–75, 2017.

[9] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[10] P. Zhang, C. Wang, N. Kumar, W. Zhang, and L. Liu, "Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[11] A. Tootoonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, and S. Shenker, "{ResQ}: Enabling {SLOs} in network function virtualization," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 283–297.

[12] H. Yu, Z. Zheng, J. Shen, C. Miao, C. Sun, H. Hu, J. Bi, J. Wu, and J. Wang, "¡monospace¿octans¡/monospace¿: Optimal placement of service function chains in many-core systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2202–2215, 2021.

[13] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664–676, 2017.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[16] T. Aladwani, "Impact of selecting virtual machine with least load on tasks scheduling algorithms in cloud computing," in *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, 2017, pp. 1–7.

[17] C. Li, J. Tang, T. Ma, X. Yang, and Y. Luo, "Load balance based workflow job scheduling algorithm in distributed cloud," *Journal of Network and Computer Applications*, vol. 152, p. 102518, 2020.

[18] H. Cui and F. You, "User-centric resource scheduling for dual-connectivity communications," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3659–3663, 2021.