You Calculate and I Provision: A DRL-assisted Service Framework to Realize Distributed and Tenant-driven Virtual Network Slicing

Xu Zhang, Baojia Li, Jianquan Peng, Xiaoqin Pan, Zuqing Zhu, *Senior Member, IEEE,*

***Abstract*—This paper studies the problem of virtual network (VNT) slicing in datacenter interconnections (DCIs), and proposes a novel service framework to better balance the tradeoff between cost-effectiveness and time-efficiency. Our idea is to partition a DCI into non-overlapped subgraphs, divide the VNT slicing in each subgraph into four collaborative steps, and get tenants involved in the calculation of virtual network embedding (VNE) schemes. With our proposal, an agent of infrastructure provider (InP) leverages deep reinforcement learning (DRL) to price and advertise the substrate resources in one subgraph, motivates tenants to request resources in a load-balanced manner, and accepts VNE schemes from the tenants to avoid resource conflicts. Meanwhile, the tenants' task is to compute their own VNE schemes independently and distributedly according to the resource information (*i.e.*, the available resources and their prices) advertised by the agent. We first design the DRL model based on the deep deterministic policy gradient (DDPG), and develop a VNT compression method based on auto-encoder (AE) to generalize the DRL's operation. Then, we study how to resolve resource conflicts among the distributedly-calculated VNE schemes, build a conflict graph (CG) to transform the VNE selection into finding the maximum weighted independent set (MWIS) in the CG, and design a polynomial-time approximation algorithm to solve the problem. Extensive simulations confirm that compared with the centralized service framework relying solely on the InP for VNE calculation, our proposed DRL-assisted distributed framework provisions VNT requests with significantly shorter computation time and comparable blocking performance.**

***Index Terms*—Virtual network embedding (VNE), Deep reinforcement learning (DRL), Auto-encoder (AE), Distributed and parallel operation, Datacenter interconnections (DCIs).**

I. Introduction

NOWADAYS, the booming of cloud computing and other Internet-based applications has pushed the traffic among datacenters (DCs) to grow rapidly [1–3]. Hence, the architecture of DC interconnections (DCIs) is facing great challenges to support the traffic demands timely and cost-effectively [4]. This dilemma has motivated people to consider network virtualization [5]. With network virtualization, traditional Internet service providers (ISPs) evolve into two types of entities, *i.e.*, the infrastructure providers (InPs) and service providers (SPs). Specifically, SPs are the tenants to lease virtual networks (VNTs) in the pay-as-you-go manner from a substrate network (SNT) that is owned by an InP, while the InP builds VNTs on-demand with substrate resources in its SNT (*i.e.*, the resources on substrate links (SLs) and substrate nodes (SNs)) [6, 7]. If the SNT is a DCI, the InP can provide SPs with IT resources in the DCs and bandwidth resources in the optical inter-DC network dynamically and adaptively, such that time-varying demands from heterogeneous network services can be satisfied cost-efficiently [8, 9]. Hence, network virtualization in the DCI not only improves the utilization of the InP's substrate resources, but also shortens the time-to-market of the SPs' network services, *i.e.*, achieving a win-win situation.

Despite the aforementioned benefits, it is still challenging for an InP to provision VNTs cost-effectively and time-efficiently. The major difficulty comes from the fact that the key problem of network virtualization, namely, virtual network embedding (VNE), is $\mathcal{NP}$-hard [10]. This means that it would be intractable for an InP to obtain the optimal VNE solution, if there are many VNTs to slice or/and the size of its SNT is relatively large. Previous studies have designed numerous algorithms to tackle VNE in various networks with different optimization objectives [10–13], and the system prototypes for VNT slicing have also been experimentally demonstrated [14–16]. Nevertheless, most of these investigations tried to solve VNE in the centralized manner, *i.e.*, the InP calculates all the VNE schemes without any tenant participation.

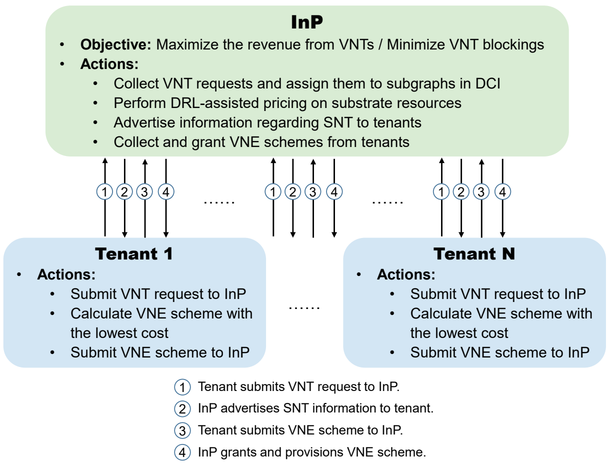
In this work, we revisit the problem of VNT slicing in DCIs, and argue that the tradeoff between cost-effectiveness and time-efficiency can be better balanced if the novel service framework shown in Fig. 1 is adopted. Here, the main idea is to get tenants involved in VNE calculation, and thus VNT slicing can be accomplished in a distributed and tenant-driven manner. Specifically, the InP still has the objective to maximize the revenue from provisioned VNTs (or to minimize VNT blockings), but it divides VNT slicing into four steps to leverage the computing power of tenants.

The InP first collects VNT requests from its tenants (**Step 1**). Then, instead of calculating the VNE schemes by itself, the InP utilizes a deep reinforcement learning (DRL) model to quickly analyze the VNT requests for resource pricing and advertising. Specifically, for each VNT request, the InP checks its characteristics with a DRL model, which will assign the request to a subgraph in the DCI to embed its VNT and price the related substrate resources accordingly (**Step 2**). Hence, in the subsequent step, the VNE schemes calculated distributedly by the tenants will have the least resource conflicts. To ensure the performance of the DRL-assisted resource pricing and

X. Zhang, B. Li, J. Peng, X. Pan and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (Email: zqzhu@ieee.org).

X. Pan is also with the Engineering Technology Center, Southwest University of Science and Technology, Mianyang, Sichuan 621010, P. R. China.

Manuscript received May 11, 2020.



InP
Objective: Maximize the revenue from VNTs / Minimize VNT blockings
Actions:
Collect VNT requests and assign them to subgraphs in DCI
Perform DRL-assisted pricing on substrate resources
Advertise information regarding SNT to tenants
Collect and grant VNE schemes from tenants
Tenant 1
Actions:
Submit VNT request to InP
Calculate VNE scheme with the lowest cost
Submit VNE scheme to InP
Tenant N
Actions:
Submit VNT request to InP
Calculate VNE scheme with the lowest cost
Submit VNE scheme to InP
① Tenant submits VNT request to InP.
② InP advertises SNT information to tenant.
③ Tenant submits VNE scheme to InP.
④ InP grants and provisions VNE scheme.

Fig. 1. Service framework for distributed and tenant-driven VNT slicing.

advertising, we design the DRL model based on the deep deterministic policy gradient (DDPG) [17], which consists of pricing and management modules. Meanwhile, since each VNT request can only be represented by a complex and high-dimensional data structure (*i.e.*, the topology and resource demands of virtual links (VLs) and virtual nodes (VNs)), we consider how to avoid the "curse of dimensionality" and generalize the operation of the DRL model. Specifically, we leverage the idea of auto-encoder (AE) [18] to propose a VNT compression method that can discard irrelevant information, reduce the dimensionality of VNTs' data, and effectively extract the key features of VNTs to feed into the DRL model.

In **Step 3**, the InP advertises the information regarding its SNT (*i.e.*, for each tenant that submitted a VNT request, the InP provides the topology and resource prices of the assigned subgraph), and lets the tenants calculate their VNE schemes independently and distributedly[1]. If two or more tenants are assigned to a same subgraph, they calculate the VNE schemes of their VNTs with the same substrate topology and resource prices. Finally, the InP collects VNE schemes from the tenants, and grants them in the way such that the revenue from VNTs can be maximized or the VNT blockings can be minimized (**Step 4**). As the VNE schemes are calculated distributedly, they could have resource conflicts, *i.e.*, provisioning multiple VNE schemes simultaneously could violate the resource capacity constraint(s) of certain SL(s)/SN(s). Hence, we study how to resolve the resource conflicts by provisioning VNE schemes selectively. Specifically, we construct a conflict graph (CG) based on the VNE schemes, transform the problem into finding the maximum weighted independent set in the CG, and design a polynomial-time approximation algorithm to solve it.

In all, our proposal of the distributed and tenant-driven service framework in Fig. 1 effectively simplifies the InP's network control and management (NC&M) and greatly improves the time-efficiency of VNT slicing in DCIs. We conduct extensive simulations to evaluate its performance, and demonstrate that compared with the centralized benchmark, it provides comparable VNT blocking probability but much better scalability in terms of running time. In summary, the major contributions of this work are

- We propose a DRL-assisted service framework for distributed and tenant-driven VNT slicing in DCIs.
- We design a DRL model, which can quickly analyze VNT requests to price and advertise the substrate resources in a DCI, for intelligently guiding the subsequent distributed VNE calculation to minimize resource conflicts.
- We develop a VNT compression method based on AE to not only limit the DRL model's input dimensionality but also generalize its operation.
- We leverage a polynomial-time approximation algorithm to select conflict-free VNE schemes to provision.

The rest of paper is organized as follows. Section II briefly surveys the related work. We describe the overall design of the proposed service framework and the concerned problem of VNT slicing in a DCI in Section III. Then, Sections IV and V elaborate on the DRL model for resource pricing and advertising and the approximation algorithm for selecting conflict-free VNE schemes, respectively. Next, we discuss the numerical simulations for performance evaluation in Section VI. Finally, Section VII summaries the paper.

II. Related Work

As the key problem of network virtualization, VNE has been studied intensively in both theoretical and experimental aspects. Using various networks as SNT, the studies in [9, 10, 12, 13, 19–23] formulated a few integer linear programming (ILP) and mixed integer linear programming (MILP) models for VNE. As the ILP/MILP models will become intractable for large-scale problems, they also designed polynomial-time heuristics and approximation algorithms to trade solution performance for time-efficiency. Through the process, it is found that the performance of VNE can be improved by considering the topology information of SNT and VNTs. For instance, inspired by PageRank [24], the investigations in [7, 25] abstracted the topology information of SNT and VNTs and utilized the abstracted information to assist node mapping. Different from the one in packet networks, the VNE in optical networks (*i.e.*, the SNT is a fixed-grid wavelength-division multiplexing (WDM) network [9] or a flexible-grid elastic optical network (EON) [26–28]) needs to solve the famous routing and spectrum assignment (RSA) problem in link mapping. Since RSA itself is $\mathcal{NP}$-hard, such VNE problems are intrinsically more complex. A relatively comprehensive survey on the existing VNE algorithms can be found in [11].

In addition to theoretical studies, people have also developed network virtualization hypervisors (NVHs) to demonstrate slicing VNTs over a shared SNT experimentally [14–16]. However, all the aforementioned studies relied solely on computing power of the InP to optimize the VNE schemes of VNT requests, *i.e.*, utilizing the purely centralized scenario. Note that, there could be many VNTs for the InP to slice [1], and in this situation, the purely centralized scenario can hardly deliver high-performance VNE solutions while maintaining reasonably good scalability in terms of computation time.

[1]Note that, our service framework does not restrict the VNE algorithm used by the tenants. In other words, with the adaptivity provided by the DRL model, it works well as long as the VNE algorithm provides the tenants with feasible VNE schemes and tries to reduce their costs.

Previously, researchers also designed algorithms to achieve distributed/parallel VNE [29–31], which were all based on the idea to partition the SNT topology into N non-overlapped subgraphs. Then, by delegating VNT requests to the subgraphs, the InP can use N processes to calculate the VNE schemes (*i.e.*, each is within a subgraph) in parallel. Although we also divide the SNT topology into subgraphs, our proposal is different from theirs, because they still let the InP calculate all the VNE schemes without any tenant participation. Moreover, as their degree of parallelism is just the number of subgraphs partitioned from the SNT (*i.e.*, N), there is a tradeoff between the degree of parallelism and the performance of parallel VNE. Specifically, the average size of non-overlapped subgraphs decreases with N, which will eventually block certain VNTs.

Recently, people started to incorporate machine learning (ML) techniques in NC&M to solve complex optimizations or/and make intelligent and timely decisions [32, 33]. Among the ML techniques, DRL has been considered as a promising one because it adopts online training, which is suitable to handle dynamic network environments [34, 35]. Leveraging ML techniques to tackle VNE problems has just started to attract research interests since recently [36–38], but all the proposals were based on the centralized scenario. In [36], the authors designed an ML model to forecast whether a VNT request should be accepted for VNE calculation. The studies in [37, 38] considered how to leverage DRL to solve VNE problems directly. The RDAM algorithm proposed in [37] used novel methods to represent and update SNT information, and designed a DRL model to solve node mapping. Dolati *et al.* [38] designed DeepViNE, whose main idea is to encode the information of SNT and VNTs as two-dimensional images and then solve VNE accordingly based on Q-learning.

Although RDAM and DeepViNE could outperform a few existing VNE algorithms, using DRL to solve VNE problems directly might suffer from two issues. Firstly, it cannot avoid the "curse of dimensionality", since the sizes of state and action spaces increase quickly with the sizes of SNT/VNT topologies. Secondly, it can hardly provide a generic solution, *i.e.*, when the topologies of SNT/VNT change, the DRL model needs to be retrained or even re-architected.

Therefore, in this work, we make tenants calculate their VNE schemes with classical VNE algorithms (*i.e.*, non-ML-based ones), but only leverage DRL to optimize the resource pricing and advertising for the InP. Then, the sizes of the DRL model's state and action spaces can be well controlled. In other words, our DRL model does not directly participate in VNE calculations, but it can adjust the prices of substrate resources intelligently to affect VNE results indirectly. We proposed the preliminary version of our DRL-assisted service framework in [39]. However, the DRL model was based on Q-learning and not optimized, and the conflict-free selection of VNE schemes was not addressed. Then, in [40], we redesigned the DRL model based on DDPG and developed a heuristic to select conflict-free VNE schemes greedily. Nevertheless, as we will show later, the DRL model can be further optimized by introducing the AE-based VNT compression. Moreover, to ensure the performance of solutions, we need a polynomial-time approximation algorithm for conflict-free VNE selections.

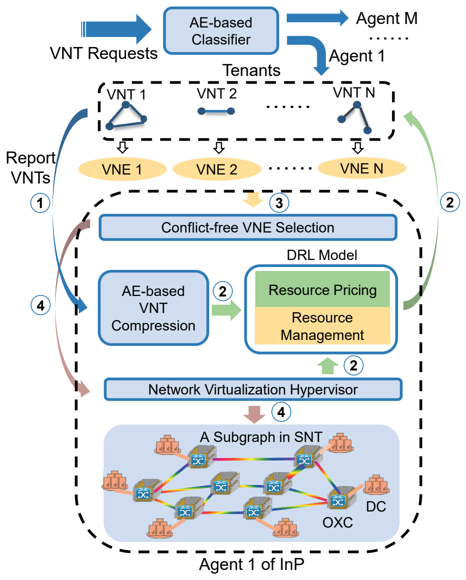


Fig. 2. Architecture of our proposed service framework.

III. Problem Description

In this section, we first describe the overall design of our service framework, and then define the network model and optimization problem of slicing VNTs in a DCI.

A. DRL-assisted Service Framework

Fig. 2 shows the architecture of our DRL-assisted service framework to realize distributed and tenant-driven VNT slicing. To ensure the scalability of our approach, we also assume that the InP can divide the topology of the DCI (*i.e.*, the SNT topology) into several non-overlapping subgraphs, if the SNT topology is relatively large [29–31]. Note that, the topology partition can be accomplished with the multilevel recursive bisection partitioning algorithm developed in [41], which can quickly partition a relatively large topology into roughly equal-sized subgraphs with three phases, for coarsening, partitioning, and un-coarsening, respectively. Moreover, the algorithm also minimizes the number of links among the obtained subgraphs. The VNT slicing in the subgraphs is managed by independent agents, which include DRL models with similar architectures and operation principles. After receiving a batch of VNT requests, the InP first uses an auto-encoder (AE) based classifier to distribute them to the agents. The AE-based classifier tries to send VNTs with similar characteristics to different agents, such that in subsequent steps, the resource conflicts among the VNE schemes obtained by the tenants can be reduced. The AE-based classifier will be designed in Section IV.

Next, each agent accomplishes the VNT slicing in its subgraph with four steps, and at each provisioning time, the operations are as follows. The tenants assigned to the agent first report all the pending VNT requests, which get stored and processed by the AE-based VNT compression module (**Step 1**), which extracts key features of the VNT requests and feeds them into the DRL model that consists of two modules, and meanwhile, the modules collect the current state of the

SNT from the NVH. Based the output of the AE-based VNT compression module, the resource pricing and management modules price the substrate resources in the agent's subgraph and turn off some resources for certain tenants, respectively, and they then send results to the tenants accordingly (**Step 2**). Specifically, the pricing module tries to guide the tenants to not use bottleneck resources, while the management module aims to shutting down different resources for the tenants to relieve resource competition. They are both based on DRL, where their DRL agents observe the same network environment, but their training and operations are independent of each other. The details regarding them will be discussed in Section IV.

Upon receiving the results on resource pricing and advertising modules, each tenant calculates its own VNE scheme independently and distributedly, with the objective of minimizing the total VNE resource cost. Then, the tenants submit their VNE schemes to the agent (**Step 3**). The agent utilizes a conflict-free VNE selection algorithm to process the VNE schemes, chooses to provision those that lead to maximized revenue from VNTs or minimized VNT blockings, and finally deploys the selected VNE schemes with the NVH (**Step 4**).

TABLE I
NOTATIONS FOR THE NETWORK MODEL

Notation	Explanation
DCI:	
v_i^E	the i-th edge SN in V^E
v_i^I	the i-th intermediate SN in V^I
$E(v)$	the set of SLs that connect to SN $v \in V$
R_i^c	the amount of available IT resources in the DC of v_i^E
R_e^b	the available bandwidth on an SL $e \in E$
k-th VNT Request $G_k^r(V_k^r, E_k^r)$:	
$v_{k,i}^r$	the i-th VN in V_k^r
$R_{k,i}^r$	the amount of IT resources demanded by VN $v_{k,i}^r$
$R_{k,e}^r$	the amount of bandwidth required by VL $e \in E_k^r$

Note that, despite the regulations of the resource pricing and management modules, the tenants can still obtain VNE schemes with resource conflicts. Therefore, the agent has to reject certain VNE schemes, which is the downside of distributed VNE calculation. From the agent's perspective, it can reduce resource conflicts by motivating the tenants to demand for substrate resources in the load-balanced way, and this can be done by pricing substrate resources adaptively and restricting the resource usage of each tenant intelligently. This is essentially a repeated leader-follower game, where the agent of the InP is the leader and the tenants are the followers. The agent first makes the decision on resource pricing and advertising to affect the tenants' behaviors (*i.e.*, preventing them from competing for scarce substrate resources). Then, the tenants independently calculate their VNE schemes whose resource conflicts will in turn affect the agent's strategy at the next provisioning time. Due to the complexity of the VNE problem and the dynamic nature of VNT requests, it would be rather difficult to analyze the game analytically. Hence, we leverage DRL to make wise decisions for the agent.

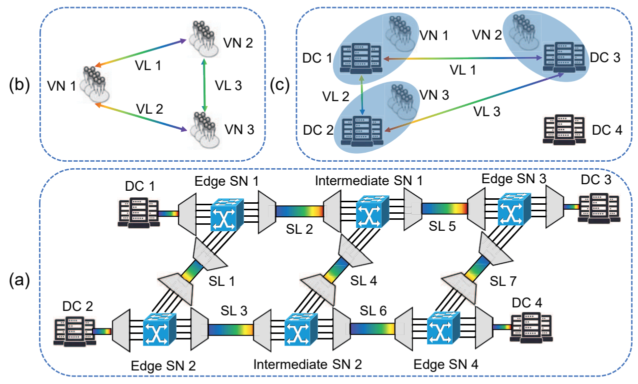


Fig. 3. Example on VNT slicing in a subgraph of DCI, (a) the SNT, (b) a tenant's VNT request, and (c) the VNE scheme computed by the tenant.

B. Network Model

Table I lists the notations defined for the network model.

1) Substrate Network: The topology of a subgraph in the SNT is modeled as a graph $G(V, E)$, where V and E are the sets of SNs and SLs in it, respectively. As shown in Fig. 3(a), there are actually two types of SNs, *i.e.*, the edge and intermediate SNs, respectively. Specifically, each edge SN includes a local DC and an optical cross-connect (OXC) and it is included in set V^E, while an intermediate SN only contains an OXC and it is enclosed in set V^I. Hence, we have $V^E \cap V^I = \emptyset$ and $V^E \cup V^I = V$. The VNT slicing considers two types of substrate resources, which are the IT resources on DCs and the bandwidth resources on SLs (*i.e.*, fiber links). In this work, we assume that the bandwidth resources can be allocated in a granularity that is much smaller than a wavelength channel, which is actually the practical case in DCIs considering the various bandwidth demands of network services [1]. In other words, we model each SL as a bandwidth pipe and tackle the VNE problem in the packet layer, and thus there is no need to consider the RSA problem in link mapping.

2) Virtual Networks: At each provisioning time, the InP could use the AE-based classifier to allocate K pending VNT requests from tenants to an agent. The k-th VNT request is denoted as $G_k^r(V_k^r, E_k^r)$, where V_k^r and E_k^r are the sets of VNs and VLs, respectively. Fig. 3(b) shows an example of VNT.

3) VNE Calculation: Based on the information provided by the agent, each tenant calculates the VNE scheme of its VNT request and tries to minimize the total VNE resource cost. Specifically, in order to solve the VNE problem, the tenant needs to a) choose a DC node in the SNT to embed each VN in its VNT such that the VN's IT resource demand can be satisfied (*i.e.*, the node mapping), and b) set up each VL between a VN pair on a substrate path with sufficient bandwidth end-to-end (*i.e.*, the link mapping). Fig. 3(c) shows the VNE scheme computed by a tenant.

As the VNE problem is $\mathcal{NP}$-hard, tenants should use heuristic algorithms to tackle it, such that cost-effective VNE schemes can be calculated time-efficiently. In Section VI, we will show that our service framework is agnostic to the VNE algorithm used by the tenant. Hence, we will not specify a heuristic for the tenants, and assume that they can leverage any existing approach with reasonably good performance.

IV. DRL-ASSISTED RESOURCE PRICING AND ADVERTISING

As shown in Fig. 2, the VNT requests from tenants are first distributed to different agents of the InP by the AE-based classifier, and then each agent processes its VNT requests with the help from the tenants. Specifically, there are three major components in each agent, *i.e.*, the DRL-based resource pricing and management modules, the AE-based VNT compression, and the conflict-free VNE selection. In this section, we explain our designs of the first two, and because the DRL-based modules define the output as well as operation principle of the AE-based VNT compression, they are introduced first. Meanwhile, as the AE-based classifier in Fig. 2 shares the same operation principle with the AE-based VNT compression, it is also briefly covered in this section.

A. Resource Pricing and Management Modules

Each agent uses the resource pricing and management modules to price the substrate resources in its subgraph of the SNT and choose the policy to hide some resources from certain tenants, respectively. Both modules are based on DRL, where their DRL agents observe the same network environment (*i.e.*, the resource utilization in the subgraph and the information regarding VNT requests), but use different action spaces.

1) Actions of Resource Pricing Module: To restrict the output dimensionality of its DRL model, we need to carefully design the actions of the pricing module. Intuitively, its actions are to price the resources on $|V^E| + |E|$ substrate elements, where the number of SLs ($|E|$) is normally much greater than that of edge SNs ($|V^E|$) in a mesh subgraph.

Definition 1. *In an SNT, we define a* ***substrate cluster (SC)*** *as a tree-type structure, which is rooted in an edge SN and may also have branches that include SLs and intermediate SNs.*

We obtain SCs by partitioning the SNT topology (as shown in Fig. 4(a)). SCs do not overlap with each other, and an SL between two SNs is randomly assigned to one of their SCs. Hence, an SC might only contain one SN but zero SL. To optimize the actions of the pricing module, we make it change the prices of the substrate resources in each SC simultaneously. This is because the resource usages of the substrate elements in an SC are normally highly-correlated [7]. Hence, we reduce the DRL's outputs from $|V^E| + |E|$ to $|V^E|$. Then, the actual pricing model is defined as follows. We first normalize the resource capacity of each substrate element (*i.e.*, an SN/SL) to one, and then let the DRL price the unit usage (*e.g.*, 1%) of the resources on each substrate element in an SC as

$$w_j = c_j + a_j \cdot \delta_j, \tag{1}$$

where j is the global index of the substrate element, c_j and δ_j are the empirically-selected constants to denote its base and incremental resource costs, respectively, and $a_j \in [0, 1]$ is a ratio whose value is determined by the DRL's actions. Hence, the DRL changes the unit price[2] of the resources on a substrate element within $[c_j, c_j + \delta_j]$. Note that, the values of c_j and δ_j can be different for different substrate elements, while for all the substrate elements in an SC, their a_j are the same.

Hence, at provision time t, the pricing module's action is to determine the ratios for all the SCs, *i.e.*, getting $\mathcal{A}_t = \{a_j, \ \forall j\}$. Meanwhile, if we denote the current network state regarding both the SNT's subgraph and the pending VNT requests assigned to use the subgraph as $\mathcal{S}_t$, the DRL of the pricing module needs to learn the pricing strategy $\pi(\mathcal{A}_t|\mathcal{S}_t)$.

2) Actions of Resource Management Module: To avoid the tenants competing for substrate resources when calculating their VNE schemes distributedly and independently, we introduce the resource management module to hide some resources from being advertised to certain tenants [42]. Specifically, we define a parameter p to denote the probability that a DC should be hidden in the resource advertisements to tenants. In each service provisioning, p uses the same value for all the tenants, but because the DCs are hidden in the resource advertisement to each tenant randomly with a probability of p, the resource advertisements to different tenants can hide different DCs, *i.e.*, the topology information received by the tenants can be different. If a DC is hidden for certain tenants, we set the amount of available IT resources on the corresponding edge SN as 0, while the tenants still receive a connected subgraph. Hence, at provision time t, the management module's action is to select a proper value for p (*i.e.*, $\mathcal{P}_t = \{p, \ p \in [0, 1]\}$), and it needs to learn the strategy $\pi(\mathcal{P}_t|\mathcal{S}_t)$.

B. Design of DRL Model

The principle of DRL is about letting an intelligent agent[3] interact with a dynamic environment and choose proper actions to address different environment states, such that the agent's reward from the environment can be maximized [43]. Hence, each DRL model involves four basic elements, *i.e.*, the agent, state, action, and reward. We design the DRL model for the resource pricing and management modules based on DDPG [17], which is known to be powerful for optimizing actions with continuous values to tackle high-dimensional states. For the two modules, their DRL models use the same neural network structure, observe the same network state, and collect the same reward. The only difference is that their action spaces are defined differently. Therefore, we use the resource pricing module as an example to explain the DRL models' design.

Fig. 4(b) shows the operation principle of the DRL model for resource pricing. The environment is just a subgraph in the SNT, and thus its state includes the information about both the subgraph (*i.e.*, the topology and resource usage on each substrate element) and the pending VNT requests. Note that, to generalize the DRL's operation, we abstract the information of VNT requests with AE-based VNT compression, which will be introduced in the next subsection. The DRL model consists of two neural networks, which are the actor neural network (A-NN) and the critic neural network (C-NN), and leverages them to optimize its decision making. Specifically, at each t, the A-

[2]Here, the "price" might not be the real unit price of substrate resources. It should be understood as a weight of the resources, which is provided by the InP to the tenants to guide their VNE calculations, such that the overall blocking probability of VNT requests can be minimized.

[3]The agent here refers to the DRL agent in each DRL model, but not an agent of the InP for network slicing.

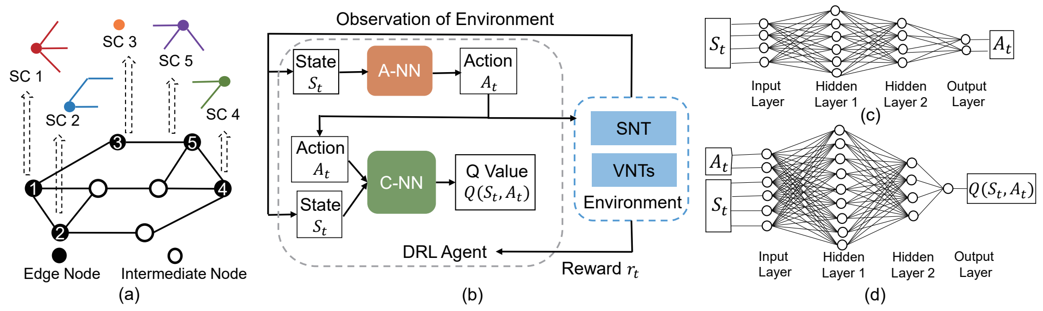


Fig. 4. (a) Example on partitioning an SNT into SCs, (b) DDPG-based DRL model for resource pricing and advertising, and (c) and (d) Architectures of A-NN and C-NN, respectively.

NN selects an action $\mathcal{A}_t$ based on its learned pricing strategy $\pi(\mathcal{A}_t|\mathcal{S}_t)$, while the C-NN evaluates the selected action.

As the action space is continuous (*i.e.*, each $a_j \in \mathcal{A}_t$ is a real number), the A-NN gets a deterministic action with the pricing function $\mathcal{A}_t = \mu(\mathcal{S}_t|\theta^a)$, where θ^a denotes its parameters. After applying action $\mathcal{A}_t$ to the environment, the DRL agent gets an instant reward r_t to evaluate the performance of $\mathcal{A}_t$ in state $\mathcal{S}_t$. Here, r_t is defined as the acceptance ratio of the VNT requests considered at t. Then, the environment proceeds to state $\mathcal{S}_{t+1}$, and the aforementioned procedure gets repeated. At each t, the agent gets a tuple $<\mathcal{S}_t, \mathcal{A}_t, r_t, \mathcal{S}_{t+1}>$ and stores it as an entry of experience, which is used in the online training of A-NN and C-NN to update their parameters (θ^a and θ^c).

The C-NN uses a value function $Q(\mathcal{S}_t, \mathcal{A}_t|\theta^c)$ to evaluate the actions taken by the A-NN, which is also optimized in online training. The training happens when the agent accumulates enough entries of experience, and the C-NN replays the entries to get the long-term reward at state $\mathcal{S}_t$

$$\mathcal{R}_t = Q(\mathcal{S}_t, \mathcal{A}_t|\theta^c). \tag{2}$$

Meanwhile, with the instant reward r_t stored in the corresponding entry, we can get another estimated long-term reward using the Bellman equation

$$\mathcal{R}'_t = r_t + \gamma \cdot Q(\mathcal{S}_{t+1}, \mathcal{A}'_{t+1}|\theta^c), \tag{3}$$

where $\mathcal{A}'_{t+1}$ is the new action provided by the updated A-NN with $\mathcal{S}_{t+1}$ as the input, and γ is the discount factor, which is a constant within $[0, 1]$. As the C-NN's training is to learn how to evaluate the A-NN's actions more accurately, its loss function is defined as

$$\sigma = \frac{1}{N}\sum_t (\mathcal{R}'_t - \mathcal{R}_t)^2, \tag{4}$$

where N is the number of experience entries used in the training. The training updates the C-NN's parameters θ^c to minimize the loss σ. In the meantime, the A-NN's training continuously optimizes the parameters θ^a of its pricing function $\mu(\mathcal{S}_t|\theta^a)$, according to the evaluation provided by the C-NN. The A-NN updates θ^a with the sampled policy gradient

$$\nabla_{\theta^a} \approx \frac{1}{N} \cdot \left[\nabla_{\mathcal{A}} Q(\mathcal{S}, \mathcal{A}|\theta^c)|_{\mathcal{S}=\mathcal{S}_t, \mathcal{A}=\mathcal{A}'_t}\right] \cdot \nabla_{\theta^a} \mathcal{A}'_t. \tag{5}$$

where $\mathcal{A}'_t$ is new action with $\mathcal{S}_t$.

Figs. 4(c) and 4(d) lay out the architectures of the A-NN and C-NN, respectively. Here, the A-NN takes the state $\mathcal{S}_t$ observed from the environment as the input, and outputs an action $\mathcal{A}_t$ to price the substrate resources in each SC. Hence, the dimensions of the A-NN's input and output layers are $|V^E|+|E|+|F_R|$ and $|V^E|$, respectively, where $|F_R|$ refers to the number of features that the AE-based VNT compression obtains from VNTs. Meanwhile, we allocate two hidden layers in the A-NN, to map its inputs to outputs (*i.e.*, $\mathcal{S}_t \rightarrow \mathcal{A}_t$) with fully-connected neural networks. The structure of the C-NN is similar to that of the A-NN, but the dimensions of its input and output layers are $2 \cdot |V^E| + |E| + |F_R|$ and 1, respectively. The C-NN's input layer takes both the state $\mathcal{S}_t$ and the action $\mathcal{A}_t$ provided by the A-NN, while its output layer is one-dimensional, which represents the evaluation of the action $\mathcal{A}_t$ in the state $\mathcal{S}_t$ (*i.e.*, $Q(\mathcal{S}_t, \mathcal{A}_t|\theta^c)$).

The design of the DRL model for the resource management module is similar, except that it utilizes the A-NN and C-NN to learn the resource shutting down strategy $\pi(\mathcal{P}_t|\mathcal{S}_t)$ in online training. As the operations of the resource pricing and management modules are correlated, we train them alternately, *i.e.*, the parameters of one module keep unchanged when those of the other one are being updated in online training.

C. AE-based VNT Compression

As the environmental state $\mathcal{S}_t$ includes the information about both the subgraph and pending VNT requests, the state space would be extremely large if we directly put the VNT requests in $\mathcal{S}_t$. This will make the DRL models converge very slowly or even not converge in training, and thus would seriously affect the performance of our service framework. Therefore, we design a VNT compression method by leveraging AE [18].

Fig. 5(a) explains the principle of AE, which uses self-supervised learning to characterize high-dimensional data, with an encoder and a decoder. The encoder maps the input samples $\mathcal{X}$ to feature space $\mathcal{Z}$ with an encoding function $g : \mathcal{X} \rightarrow \mathcal{Z}$, while the decoder reverts the process to obtain the reconstructed samples $\mathcal{X}'$ based on $\mathcal{Z}$ with a decoding function $f : \mathcal{Z} \rightarrow \mathcal{X}'$. To optimize the encoder/decoder, AE leverages a DNN to minimize the reconstruction errors between $\mathcal{X}$ and $\mathcal{X}'$. Although AE has already demonstrated its effectiveness in the compression and reconstruction of Euclidean data, such as images, texts and videos, we are facing the following two challenges when trying to utilize it for VNT compression.

Firstly, the high-dimensional graph-structured data regarding VNT requests is non-Euclidean and irregular. For instance,

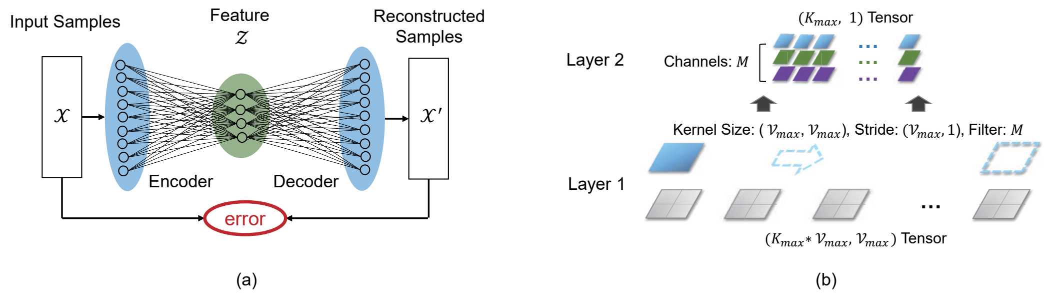

Input Samples
Feature $\mathcal{Z}$
Reconstructed Samples
$\mathcal{X}$
$\mathcal{X}'$
Encoder
Decoder
error
(a)
$(K_{max},\ 1)$ Tensor
Layer 2
Channels: M
Kernel Size: $(\mathcal{V}_{max}, \mathcal{V}_{max})$, Stride: $(\mathcal{V}_{max}, 1)$, Filter: M
Layer 1
$(K_{max} * \mathcal{V}_{max}, \mathcal{V}_{max})$ Tensor
(b)

Fig. 5. (a) Generic principle of AE, and (b) AE-based VNT compression.

each VNT has a variable number of unordered VNs, and the VNs can have different numbers of adjacent nodes. This would make certain important operations (*e.g.*, convolution), which can be easily done on Euclidean data, not directly applicable. Secondly, how to organize the unordered VNTs and input them in the DNN of AE is also challenging. Hence, inspired by the work in [44], we first design the preprocessing to transform the graph-structured data regarding VNT requests into a structure that can be processed by convolutions, and then design the DNN to extract the important features with AE.

Algorithm 1: Preprocessing of VNT Requests

```
   Input: $K$ VNT requests $\{G_k^r(V_k^r, E_k^r)\}$, $K_{\max}$, $\mathcal{V}_{\max}$.
   Output: A list of regular Euclidean data $\mathcal{G}_E$.
1  $\mathcal{G}_E = \emptyset$;
2  sort VNT requests in descending order of $|V_k^r|$ (primary)
   and $|E_k^r|$ (secondary);
3  for each VNT request $G_k^r(V_k^r, E_k^r)$ in sorted order do
4      sort its VNs in descending order of node degree;
5      represent $G_k^r$ as a $|V_k^r|\times|V_k^r|$ adjacency matrix $\mathbf{A}_k$;
6      update $\mathbf{A}_k$ to $\mathcal{V}_{\max}\times\mathcal{V}_{\max}$ matrix with zero padding;
7      insert $\mathbf{A}_k$ in $\mathcal{G}_E$;
8  end
9  if $K < K_{\max}$ then
10     insert $(K_{\max} - K)$ $\mathcal{V}_{\max}\times\mathcal{V}_{\max}$ 0-matrices in $\mathcal{G}_E$;
11 end
```

1) Preprocessing of VNT Requests: Algorithm 1 shows the procedure of VNT preprocessing. Here, in addition to K VNT requests, the algorithm also needs to know $K_{\max}$ as the upper-bound of K and $\mathcal{V}_{\max}$ as the maximum number of VNs in a VNT. As the preprocessing is conducted by an agent of the InP, it should not be difficult to set or estimate $K_{\max}$ and $\mathcal{V}_{\max}$. The basic idea of *Algorithm* 1 is to organize the VNT requests in a fixed-sized data structure where the data of each VNT is ordered uniquely. Therefore, the feature extraction with AE would provide a unique mapping for $\mathcal{X} \rightarrow \mathcal{Z}$. The sorting of VNT requests has a time complexity of $O(K_{\max}^2)$, and the number of operations to construct an adjacency matrix is proportional to $\frac{\mathcal{V}_{\max}\cdot(\mathcal{V}_{\max}-1)}{2}$. Hence, the time complexity of *Algorithm* 1 is $O(K_{\max}^2 + K_{\max}\cdot\mathcal{V}_{\max}^2)$.

2) Design of AE for VNT Compression: Our AE for VNT compression considers the adjacency matrix of each VNT as a receptive field. *Algorithm* 1 provides us with $K_{\max}$ sparse matrices, each of which has a size of $\mathcal{V}_{\max}\times\mathcal{V}_{\max}$. The matrices can be shaped as a $(K_{\max}\cdot\mathcal{V}_{\max}, \mathcal{V}_{\max})$ tensor, which is used as the input of the AE's neural network. Specifically, the operation of the VNT compression based on AE is explained in Fig. 5(b). We first apply a 2-dimensional convolutional layer with receptive field $(\mathcal{V}_{\max}, \mathcal{V}_{\max})$ and stride $(\mathcal{V}_{\max}, 1)$. The setting on the sizes of the receptive field and stride means that the convolution kernel only moves laterally, and every time it just jumps from one adjacency matrix to another. For example, the blue square in Fig. 5(b) represents a convolution kernel, which will move from left to right and perform a convolution operation on the input tensor. As the first layer of the AE only has one input channel, the actual size of the convolution kernel is $(\mathcal{V}_{\max}, \mathcal{V}_{\max}, 1)$. The convolution kernel sweeps over the input tensor, makes multiplication summation of the matrix elements in the receptive field, and superimposes the deviation,

$$
\begin{aligned}
\mathcal{L}_o(i) &= [\mathcal{L}_i \bigoplus w](i) + b \\
&= \sum_{x=0}^{\mathcal{V}_{\max}-1} \sum_{y=0}^{\mathcal{V}_{\max}-1} [\mathcal{L}_i(\mathcal{V}_{\max}\cdot i + x, y)\cdot w(x,y)] + b, \qquad (6) \\
i &\in [0, K_{\max}-1],
\end{aligned}
$$

where $\mathcal{L}_i$ and $\mathcal{L}_o$ are the input and output of the first layer, respectively, and w and b denote the weight and bias of the convolution kernel, respectively. We obtain a $(K_{\max}, 1)$ tensor with the convolution in Eq. (6), where each element is the aggregated information of a VNT request.

The first layer of the AE uses M convolution kernels for the aforementioned operation, and thus the number of its outputs (*i.e.*, the inputs to the second layer) is M. Hence, the size of the tensor obtained by the first layer is $(K_{\max}, 1, M)$. The operation in the second layer is the same as that in a classical one-dimensional convolutional neural network. In our service framework, the AE is trained in advance to minimize reconstruction errors. Then, during operation, it compresses VNT requests before inputting them to the DRL-based resource pricing and management modules. The AE-based classifier in Fig. 2 uses the same operation principle. Specifically, it also leverages AE to extract and characterize the high-dimensional data regarding VNT requests, and will send VNTs with similar characteristics to different agents of the InP.

V. Selecting Conflict-free VNE Schemes

The AE-based VNT classification and the DRL-assisted resource pricing and advertising modules cannot eliminate resource conflicts. Therefore, each agent of the InP still needs an algorithm to select conflict-free VNE schemes to provision. The VNE selection should be done in a time-efficient and cost-effective manner, since it directly affects the performance of the InP. In this section, we transform it into a classic optimization problem, and leverage the approach in [45] to design a polynomial-time approximation algorithm to solve it.

A. Problem Definition

The VNE scheme of each VNT request $G_k^r(V_k^r, E_k^r)$ can be represented by two sets of variables, *i.e.*,

$$\mathcal{M}_k = \{x_{i,i'}^k,\ y_{e,e'}^k,\ \forall v_{k,i}^r \in V_k^r,\ e \in E_k^r\}, \tag{7}$$

where the variables are defined as:

- $x_{i,i'}^k$: the boolean variable that equals 1 if the i-th VN in V_k^r ($v_{k,i}^r$) is mapped on the i'-th edge SN in V^E ($v_{i'}^E$), and 0 otherwise.
- $y_{e,e'}^k$: the boolean variable that equals 1 if VL $e \in E_k^r$ goes through SL $e' \in E$, and 0 otherwise.

The optimization of VNE selection can be modeled as follows.

Parameters:

- w_k: the positive weight of the VNE scheme (*i.e.*, $\mathcal{M}_k$) of the k-th VNT request $G_k^r(V_k^r, E_k^r)$.

Variables:

- ξ_k: the boolean variable that equals 1 if the VNE scheme $\mathcal{M}_k$ is selected, and 0 otherwise.

Objective:

The objective of the VNE selection is to maximize the total weight of the selected VNE schemes. Here, the weight of each VNE scheme is defined by the InP to guide the optimization. For instance, if the InP defines the weights as $\{w_k = 1,\ \forall k \in [1, K]\}$, the objective will be to minimize VNT blockings.

$$\text{Maximize} \quad \sum_{k=1}^{K} \xi_k \cdot w_k. \tag{8}$$

Constraints:

$$\sum_{k=1}^{K} \xi_k \cdot \left(\sum_{v_{k,i}^r \in V_k^r} x_{i,i'}^k \cdot R_{k,i}^r \right) \leq R_{i'}^c, \quad \forall v_{i'}^E \in V^E, \tag{9}$$

$$\sum_{k=1}^{K} \xi_k \cdot \left(\sum_{e \in E_k^r} y_{e,e'}^k \cdot R_{k,e}^r \right) \leq R_{e'}^b, \quad \forall e' \in E. \tag{10}$$

Eqs. (9)-(10) ensure that the selected VNE schemes do not cause any IT/bandwidth resource conflicts.

B. Algorithm Design

With all the VNE schemes calculated by the tenants, an agent of the InP can build a conflict graph (CG) $G^c(V^c, E^c)$, where each node in set V^c corresponds to a VNE scheme $\mathcal{M}_k$ and thus associates with a weight w_k, and two nodes in V^c are connected by a link in set E^c if there are potential resource conflicts between their VNE schemes. Here, we refer to the resource conflicts as "potential" because the connected nodes in the CG might not be mutually exclusive. For instance, the VNE schemes in Fig. 6(a) suggest that embedding $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ in the SNT simultaneously will lead to a resource conflict on *SN* 8, and thus the nodes for the three VNE schemes are all connected in the CG in Fig. 6(b). Nevertheless, if we reject $\mathcal{M}_3$, there will be no resource conflict on *SN* 8, and then $\mathcal{M}_1$ and $\mathcal{M}_2$ can be provisioned simultaneously. To this end, we can see that the VNE selection problem becomes to find the maximum weighted independent set (MWIS) in the CG, after removing certain VNE schemes (*i.e.*, nodes in the CG) and the potential resource conflicts due to them (*i.e.*, links in the CG). Fig. 6(c) illustrates an example of the MWIS.

With the CG, we design *Algorithm* 2 based on the approach in [45] to solve the VNE selection problem. *Lines* 1-3 are for the initialization. In each iteration, the while-loop deletes a node from the CG in the greedy manner, and updates it from G_i^c to G_{i+1}^c (*Lines* 4-9). In *Line* 5, $w(v)$ returns the weight of the VNE scheme that is represented by node v in the CG, while $d(G_i^c, v)$ gives the degree of node v in graph G_i^c. Note that, when updating the CG in *Line* 7, we remove not only node u_i but also the potential resource conflicts due to it. For example, in the CG in Fig. 6(b), if we remove the node for $\mathcal{M}_3$, the link between the nodes for $\mathcal{M}_1$ and $\mathcal{M}_2$ should also be deleted because they will not have resource conflicts anymore. The time complexity of *Algorithm* 2 is $O(K^2 \cdot (|V^E| + |E|))$.

Algorithm 2: Selection of Conflict-free VNE Schemes

```
Input: Subgraph of SNT $G(V, E)$, VNT requests and
       their VNE schemes $\{G_k^r(V_k^r, E_k^r),\ \mathcal{M}_k,\ \forall k\}$.
Output: Index set of selected VNE schemes $K'$.
1  hypothetically embed all the VNE schemes in
   $\{\mathcal{M}_k,\ \forall k \in [1, K]\}$ to find potential resource conflicts;
2  build CG $G^c(V^c, E^c)$ based on the potential conflicts;
3  $i = 0$, $G_i^c(V_i^c, E_i^c) = G^c(V^c, E^c)$;
4  while $E_i^c \neq \emptyset$ do
5      $u_i = \underset{v \in V_i^c}{\operatorname{argmin}} \left\{ \frac{w(v)}{d(G_i^c, v) \cdot \left[ d(G_i^c, v) + 1 \right]} \right\}$;
6      delete node $u_i$ from $G_i^c(V_i^c, E_i^c)$;
7      update $G_i^c$ accordingly to obtain $G_{i+1}^c$;
8      $i = i + 1$;
9  end
10 get index set $K'$ based on $G_i^c(V_i^c, E_i^c)$;
```

Meanwhile, we hope to point out that our VNE selection is still different from the problem of finding MWIS in [45], because the conflicts in the CG are not necessarily mutually exclusive. Hence, *Algorithm* 2 can operate differently from that in [45], especially for how to update the CG in each iteration (*Lines* 6-7). More importantly, the difference makes the approximation ratio proved in [45], *i.e.*,

$$\eta = \frac{1}{d_{\max}(G^c) + 1}, \tag{11}$$

where $d_{\max}(G^c)$ returns the maximum node degree of CG G^c, only provide a relatively loose lower-bound. We can easily verify that the actual approximation ratio achieved by *Algorithm* 2 could be much higher. Also, the approximation ratio in Eq. (11) is still not a constant factor, because $d_{\max}(G^c)$

is not upper-bounded in an arbitrary CG. However, as we have

$$d_{\max}(G^c) \leq |V^c| - 1 = K - 1, \tag{12}$$

where K is the total number of VNE schemes that the agent received, we can maintain the approximation ratio of *Algorithm* 2 by limiting the value of K with an upper-bound $K_{\max}$, which is a reasonable approach to take in real NC&M.

VI. Performance Evaluation

In this section, we perform extensive simulations to evaluate the performance of our proposed service framework.

A. Simulation Setup

The simulations consider SNTs and VNTs whose sizes are various. Each VNT uses a random topology in which each VN pair is directly connected with a probability of 0.5. Meanwhile, to mimic the situation in real DCIs, we obtain the topology of an SNT by extending and combining the famous topologies for wide-area networks (*e.g.*, the NSFNET topology [26]). In each SNT, we randomly select 50% SNs as edge SNs and allocate certain IT resources in the DCs attached to them. The VNT requests are generated dynamically with the Poisson traffic model, where the number of requests arriving in each service cycle follows the Poisson distribution with an average of $\bar{K}$, while the holding time of each request obeys the negative exponential distribution with an average of t_{ϖ} service cycles. Therefore, the traffic load of VNT requests can be quantified as $\bar{K} \cdot t_{\varpi}$. We assume that each VNT can contain $\mathcal{V}_{\max} = 6$ VNs at most, while the largest number of VNT requests that can be processed in a batch actually changes according to simulation scenarios, as $K_{\max} \in [10, 40]$. In the DRL of the pricing module, the two hidden layers of the A-NN consist of 50 and 40 neurons, respectively, while the numbers of neurons in those of the C-NN are 60 and 30, respectively. For the management module, the A-NN and C-NN use the same numbers of neurons in their hidden layers, as 50 and 20, respectively. The rest of the key parameters are listed in Table II. Simulations average the results from 5 independent runs to get each data point, for ensuring sufficient statistical accuracy.

As we have explained before, our service framework is agnostic to the actual VNE algorithm used by the tenants, due to the adaptivity provided by the DRL model. In other words, if the tenants in our distributed service framework and the InP in the centralized one use similar VNE algorithms, our service framework should provision VNT requests with significantly shorter computation time and comparable blocking performance[4]. Note that, we need to change a VNE algorithm slightly when adapting it in our service framework. Specifically, we replace the capacity of SNs/SLs used in the original algorithm with operators related to resource prices.

TABLE II
Key Simulation Parameters

Each SNT	Number of SNs	$[14, 112]$
	Number of SLs	$[22, 248]$
	IT resource capacity of an edge SN	$[20, 30]$ units
	Bandwidth resource capacity of an SL	$[20, 30]$ units
Each VNT	Number of VNs	$[2, 4]$
	Connectivity of VN pairs	0.5
	IT resource demand of a VN	$[0.5, 1]$ unit
	Bandwidth demand of a VL	$[0.2, 1]$ unit

B. Performance of Conflict-free VNE Selection

We first verify the performance of the approximation algorithm for conflict-free VNE selection. The subgraph of SNT takes the NSFNET topology (14 SNs and 22 SLs), and the tenants use a modified version of the global resource capacity (GRC) based VNE (GRC-VNE) in [7]. To select conflict-free VNE schemes from those provided by the tenants, we first solve the ILP model defined by Eqs. (7)-(10) to get exact solutions, and then obtain approximation solutions with *Algorithm* 2. Fig. 7 compares the solutions from the ILP and approximation algorithm, which indicates that the latter approximates the exact solutions well. Meanwhile, the running

[4]The simulations assume that the InP's objective is to minimize VNT blockings, *i.e.*, we have $\{w_k = 1, \ \forall k\}$ in Eq. (8). We also confirm that the service framework performs similarly even if $\{w_k\}$ take different values.

TABLE III
Results on Running Time (seconds)

# of VNE Schemes	10	15	20	25
ILP	0.00012	0.00292	0.00332	2.2265
Approximation Algorithm	0.00019	0.00031	0.00035	0.00072
# of VNE Schemes	**30**	**35**	**40**	
ILP	17.575	1519.7	26289	
Approximation Algorithm	0.00116	0.00352	0.00843	

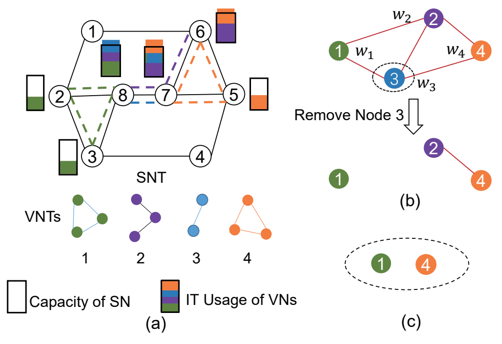


Fig. 6. Example on solving VNE selection, (a) SNT, VNTs and VNE schemes, (b) building CG for VNE schemes, and (c) finding MWIS in CG.

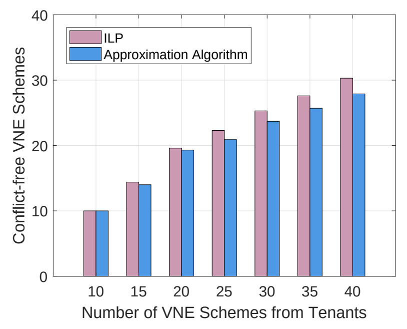


Fig. 7. Performance of conflict-free VNE selection.

time of the algorithms is listed in Table III. We can see that approximation algorithm is much more time-efficient.

C. Benchmarking with Centralized VNE Algorithms

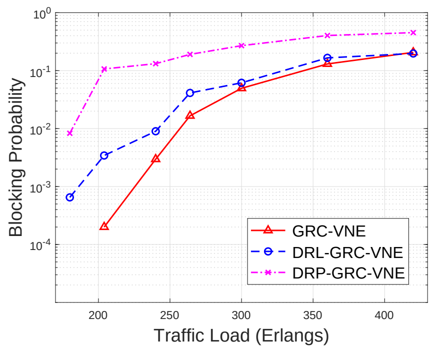
Next, we perform dynamic simulations to compare our distributed service framework with the centralized one.

1) Sensitivity to Tenants' VNE Algorithm: We consider an SNT topology that includes 56 SNs, and allocate 4 agents to manage the DCI. To test our framework's sensitivity to tenants' VNE algorithm, we consider two VNE algorithms, *i.e.*, the GRC-VNE [7] and improved closeness (IC) based VNE (IC-VNE) [46]. Using each VNE algorithm, the simulations compare the centralized framework with our proposed one. Specifically, the InP in the centralized framework use the algorithm to calculate VNE schemes for all the VNT requests, while in our proposal, the tenants use a slightly modified version of the algorithm to obtain their own VNE scheme independently. We put "DRL-" in front of an algorithm's name to refer to its usage in our DRL-assisted distributed framework, *e.g.*, "DRL-GRC-VNE". To justify the effectiveness of our DRL models, we also consider a framework that uses the same VNT provisioning procedure as our DRL-assisted distributed framework but replaces the DRL models with a deterministic algorithm for adjusting the policy of resource pricing and advertising. Specifically, the algorithm makes the unit price of the resources on each substrate element decrease in proportional with the resources' availability, and fixes the probability of hiding a DC in resource advertising. For fair comparisons, we adjust the settings of the deterministic algorithm to minimize its blocking probability, and it is referred to as deterministic resource pricing (DRP). We put "DRP-" in front of a VNE algorithm's name if it is used with the DRP algorithm.

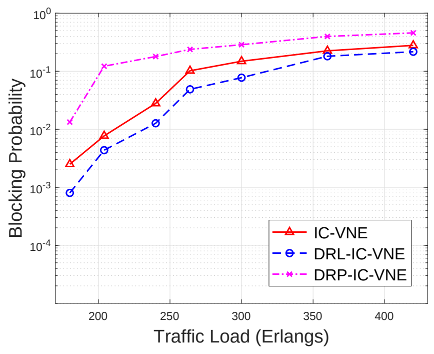
Fig. 8 shows the results on blocking probability. In Fig. 8(a), we observe that the blocking probability from DRL-GRC-VNE is higher than that from GRC-VNE, which can be explained as follows. GRC-VNE is good at distributing VNTs evenly in the SNT with the help of the global resource information, and thus the blocking performance of DRL-GRC-VNE can be slightly worse due to the occasional resource conflicts among distributed-calculated VNE schemes from tenants. The results in Fig. 8(b) indicate that DRL-IC-VNE and IC-VNE have similar blocking performance, but this time, the blocking probability from DRL-IC-VNE is lower than that from IC-VNE. This is because IC-VNE cannot distribute VNTs evenly in the global manner, while the resource pricing in our DRL-assisted framework can overcome this issue. Therefore, the disadvantage due to resource conflicts gets compensated. We also notice that with both VNE algorithms, the DRP-based framework performs much worse than our DRL-assisted ones. This actually confirms the effectiveness of our DRL models, *i.e.*, they can price and advertise resources intelligently, such that the tenants are motivated to ask for substrate resources in the way that resource conflicts can be minimized.

Table IV lists the ratio of the running time of the centralized framework to that of our distributed framework. For fair comparisons, the running time of our distributed framework covers the time taken by its overall procedure (*i.e.*, delegating VNT requests to the subgraphs and **Steps 1-4** in Fig. 2). With both VNE algorithms, our distributed framework is much more time-efficient than the centralized one, and the advantage becomes more significant as the traffic load increases. This is because the centralized framework lets the InP calculate VNE schemes for all the VNT requests, while our distributed framework makes tenants compute their own VNE schemes independently and in parallel (*i.e.*, in **Step 3**). Meanwhile, with IC-VNE, our proposal achieves larger reduction on running time. This is because our resource pricing can distinguish edge SNs better in node mapping, and thus it reduces the number of candidate edge SNs to check for embedding each VN.

Furthermore, to confirm that our DRL-assisted distributed framework operates more stably, we fix the traffic load at 264 Erlangs and plot the how the blocking probability changes over time in the centralized and distributed frameworks in Fig. 9. For both VNE algorithms, the instant blocking rate from our framework changes within much smaller ranges than that from the centralized framework, even though the two frameworks perform similarly in terms of long-term blocking probability. We also try different traffic loads and can always observe similar trends in the results. These results further verifies the adaptivity of our DRL-based approach.



(a) GRC-VNE

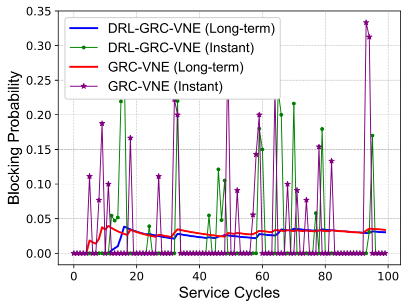


(b) IC-VNE

Fig. 8. Blocking probability from centralized and distributed frameworks.

TABLE IV
RUNNING TIME RATIO OF CENTRALIZED VERSUS DISTRIBUTED

Traffic Load (Erlangs)	204	264	300	360	420
GRC-VNE/DRL-GRC-VNE	10.5	13.5	14.6	16.9	19.7
IC-VNE/DRL-IC-VNE	36.2	42.0	44.6	55.8	60.9


(a) GRC-VNE

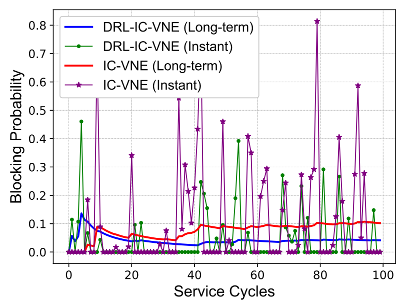
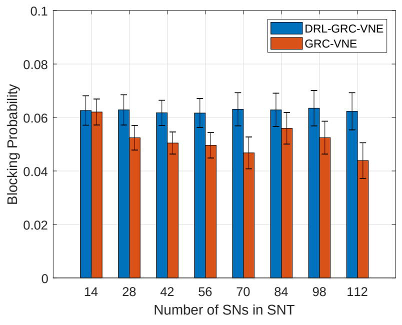

(b) IC-VNE

Fig. 9. Blocking probability over time when traffic load is 264 Erlangs.


(a) GRC-VNE

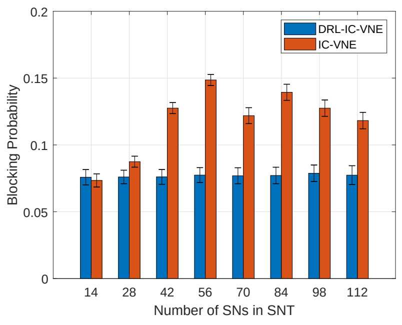

(b) IC-VNE

Fig. 10. Blocking probability in SNTs with different sizes.

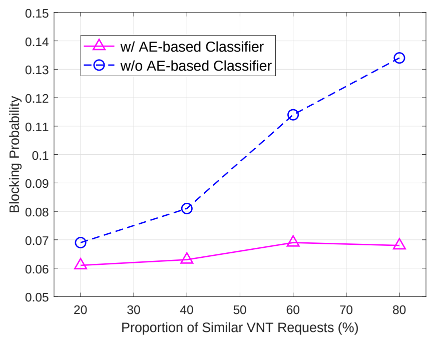


Fig. 11. Performance comparison of DRL-GRC-VNE with and without AE-based classifier (56 SNs in 4 subgraphs, and traffic load at 300 Erlangs).

2) Scalability Analysis: To analyze the scalability of our DRL-assisted distributed framework, we change the size of the SNT to include 14 to 112 SNs and increase the traffic load proportionally (*i.e.*, from 75 to 600 Erlangs), while the number of agents changes within $[1, 8]$ accordingly. Fig. 10 shows the results on blocking probability in SNTs with different sizes. We can see similar trends as those in Fig. 8, *i.e.*, GRC-VNE performs slightly better than DRL-GRC-VNE, while DRL-IC-VNE provides lower blocking probability than IC-VNE. The running time ratios in Table V suggest that for the SNTs with different sizes, our DRL-assisted service framework is still much more time-efficient, and its advantage on time-efficiency increases with the SNT's size. We also confirm that as long as the traffic load changes in proportional with the SNT's size, the trends in Fig. 10 and Table V can always be observed.

D. Effects of AE-based Classifier

We also perform simulations to verify the effectiveness of the AE-based classifier. The simulations use an SNT with 56 SNs, which is divided into 4 subgraphs, and fix the traffic load as 300 Erlangs. With this setting, the blocking probability of our DRL-GRC-VNE is a few percent in Fig. 8(a), which is in the range of practical operation. Then, we try to remove the AE-based classifier from the service framework, and change the ratio of similar VNT requests in each batch. The results on blocking probability are shown in Fig. 11, which indicates that the scheme with the AE-based classifier always outperforms the one without it, and the performance gap increases with the ratio of similar VNT requests in each batch. This confirms the effectiveness of the AE-based classifier. We also perform more simulations with different settings, and verify that the results always follow the same trend as shown in Fig. 11.

TABLE V
RUNNING TIME RATIO OF CENTRALIZED VERSUS DISTRIBUTED

Number of SNs Traffic Load (Erlangs)	14 75	42 225	70 375	98 525	112 600
GRC-VNE/DRL-GRC-VNE	4.0	9.6	21.6	49.8	70.7
IC-VNE/DRL-IC-VNE	4.0	19.5	83.7	209.8	289.2

VII. CONCLUSIONS

In this work, we proposed a DRL-assisted distributed and tenant-driven service framework to realize VNT slicing in DCIs, such that the tradeoff between cost-effectiveness and time-efficiency can be better balanced. The main idea was to get tenants involved in VNE calculation. Extensive simulations verified that compared with the centralized service framework, our proposal provisions VNT requests with significantly shorter computation time and comparable blocking performance.

Acknowledgments

This work was supported in part by the NSFC projects 61871357, 61771445 and 61701472, Zhejiang Lab Research Fund 2019LE0AB01, CAS Key Project (QYZDY-SSW-JSC003), and SPR Program of CAS (XDC02070300).

References

[1] Cisco Global Cloud Index: Forecast and Methodology, 2016-2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions /service-provider/visual-networking-index-vni/index.html

[2] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," *J. Lightw. Technol.*, vol. 33, pp. 3005–3015, Jul. 2015.

[3] X. Xie *et al.*, "Evacuate before too late: Distributed backup in inter-DC networks with progressive disasters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, pp. 1058–1074, May 2018.

[4] P. Lu *et al.*, "Highly efficient data migration and backup for Big Data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[5] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Comput.*, vol. 38, pp. 34–41, Apr. 2005.

[6] M. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, pp. 20–26, Jul. 2009.

[7] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[8] M. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 909–928, Second Quarter 2013.

[9] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[10] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 206–219, Feb. 2012.

[11] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 1888–1906, Fourth Quarter 2013.

[12] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.

[13] Y. Xue, J. Peng, K. Han, and Z. Zhu, "On table resource virtualization and network slicing in programmable data plane," *IEEE Trans. Netw. Serv. Manag., in Press*, vol. 16, pp. 647–660, Jun. 2019.

[14] R. Munoz *et al.*, "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks," *J. Opt. Commun. Netw.*, vol. 7, pp. B62–B70, Nov. 2015.

[15] J. Yin *et al.*, "Experimental demonstration of building and operating QoS-aware survivable vSD-EONs with transparent resiliency," *Opt. Express*, vol. 25, pp. 15 468–15 480, 2017.

[16] Z. Zhu *et al.*, "Build to tenants' requirements: On-demand application-driven vSD-EON slicing," *J. Opt. Commun. Netw.*, vol. 10, pp. A206–A215, Feb. 2018.

[17] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, Feb. 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[18] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, Jul. 2006.

[19] Q. Hu, Y. Wang, and X. Cao, "Resolve the virtual network embedding problem: A column generation approach," in *Proc. of INFOCOM 2013*, pp. 410–414, Apr. 2013.

[20] C. Papagianni *et al.*, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Comput.*, vol. 62, pp. 1060–1071, Jun. 2013.

[21] L. Gong, W. Zhao, Y. Wen, and Z. Zhu, "Dynamic transparent virtual network embedding over elastic optical infrastructures," in *Proc. of ICC 2013*, pp. 3466–3470, Jun. 2013.

[22] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[23] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[24] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Comput. Netw.*, vol. 56, pp. 3825–3833, Dec. 2012.

[25] S. Zhang *et al.*, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, pp. 816–827, Mar. 2013.

[26] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[27] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[28] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[29] M. Beck *et al.*, "A distributed, parallel, and generic virtual network embedding framework," in *Proc. of ICC 2013*, pp. 3471–3475, Jun. 2013.

[30] H. Cao *et al.*, "Location aware and node ranking value-assisted embedding algorithm for one-stage embedding in multiple distributed virtual network embedding," *IEEE Access*, vol. 6, pp. 78 425–78 436, 2018.

[31] A. Song *et al.*, "Distributed virtual network embedding system with historical archives and set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst., in Press*, 2019.

[32] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 393–430, First Quarter 2019.

[33] F. Musumeci *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 1383–1408, Second Quarter 2019.

[34] X. Chen *et al.*, "DeepRMSA: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *J. Lightw. Technol.*, vol. 37, pp. 4155–4163, Aug. 2019.

[35] B. Li, W. Lu, and Z. Zhu, "Deep-NFVOrch: Leveraging deep reinforcement learning to achieve adaptive vNF service chaining in EON-DCIs," *J. Opt. Commun. Netw., in Press*, 2019.

[36] A. Blenk, P. Kalmbach, P. van der Smagt, and W. Kellerer, "Boost online virtual network embedding: Using neural networks for admission control," in *Proc. of CNSM 2016*, pp. 10–18, Nov. 2016.

[37] H. Yao *et al.*, "RDAM: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding," *IEEE Trans. Emerg. Topics Comput., in Press*, 2018.

[38] M. Dolati, S. Hassanpour, M. Ghaderiy, and A. Khonsari, "DeepViNE: Virtual network embedding with deep reinforcement learning," in *Proc. of INFOCOM WKSHIPS NI 2019*, pp. 879–885, Apr. 2019.

[39] W. Lu, H. Fang, and Z. Zhu, "AI-assisted resource advertising and pricing to realize distributed tenant-driven virtual network slicing in inter-DC optical networks," in *Proc. of ONDM 2018*, pp. 1–6, May 2018.

[40] X. Zhang, W. Lu, B. Li, and Z. Zhu, "DRL-based network orchestration to realize cooperative, distributed and tenant-driven virtual network slicing," in *Proc. of ACP 2019*, pp. 1–3, Nov. 2019.

[41] G. Karypis and V. Kumar, "Parallel multilevel series k-way partitioning scheme for irregular graphs," *Siam Rev.*, vol. 41, pp. 278–300, Feb. 1999.

[42] W. Fang *et al.*, "Joint defragmentation of optical spectrum and it resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Apr. 2015.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[44] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. of NIPS 2017*, pp. 1024–1034, Dec. 2017.

[45] S. Sakai and K. Togasaki, M.and Yamazaki, "A note on greedy algorithms for the maximum weighted independent set problem," *Discrete Appl. Math.*, vol. 126, pp. 313–322, Mar. 2003.

[46] Z. Wang *et al.*, "Virtual network embedding by exploiting topological information," in *Proc. of GLOBECOM 2012*, pp. 2603–2608, Dec. 2012.