Reinforcement Learning-Based Multi-Domain Network Slice Composition with Topology Aggregation

Congzhou Li[1], Genya Ishigaki[2], Zhouxiang Wu[1], Divya Khanure[1], Riti Gour[3], and Jason P. Jue[1]
1. Department of Computer Science, The University of Texas at Dallas, Richardson, Texas 75080, USA
2. Department of Computer Science, San Jose State University, San Jose, CA 95192, USA
3. Department of Aviation and Technology, San Jose State University, San Jose, CA 95192, USA

***Abstract*—The establishment of network slices across multiple network domains requires orchestrations across domains, which is challenging due to the heterogeneity of domain-specific features and limited visibility into each domain's state information. Topology Aggregation (TA) enables domains to provide aggregated network representations in cross-domain collaborations. We propose a reinforcement learning (RL)-based framework to minimize the deployment cost of network slice provisioning across multi-domain networks. This framework takes the slice service request and aggregated network state representations encoded by Graph Neural Networks (GNNs) as input, and yields a composition plan which consists of the construction of the network slice and the placement of service functions. The simulation results show that the proposed framework enables more cost-effective slice deployment compared to a heuristic approach.**

***Index Terms*—network slicing, reinforcement learning, service function chain, topology aggregation**

I. Introduction

To facilitate the management and allocation of network resources, network slicing (NS), as a virtualization technology, divides physical network infrastructures into different virtual slices, where each slice consists of a set of virtual computing and networking resources that are capable of meeting the service requirements for specific applications [1]. In a network slice, Virtual Network Functions (VNFs) are organized as a Service Function Chain (SFC) or Forwarding Graph (FG) to serve the applications.

As applications span wider geographic areas, it is expected that network slices will involve the use of resources over multiple infrastructure domains. The problem of network slice provisioning in a multi-domain network environment involves the mapping of slices across domains and the placement of service functions to satisfy the network service requirements and computation requirements of the slice. Such planning tasks need to collect computation and network state information from each domain for calculating a cost-efficient mapping of virtual nodes within each domain and across domains that satisfies the overall service requirements. However, different domains may have heterogeneous features and complex topologies, which increase the decision space exponentially and make the global optimal slice mapping problem intractable. Another challenge is that some domains may avoid disclosing full intra-domain information regarding domain resources and topology due to scalability or privacy concerns, limiting the feasibility of global planning.

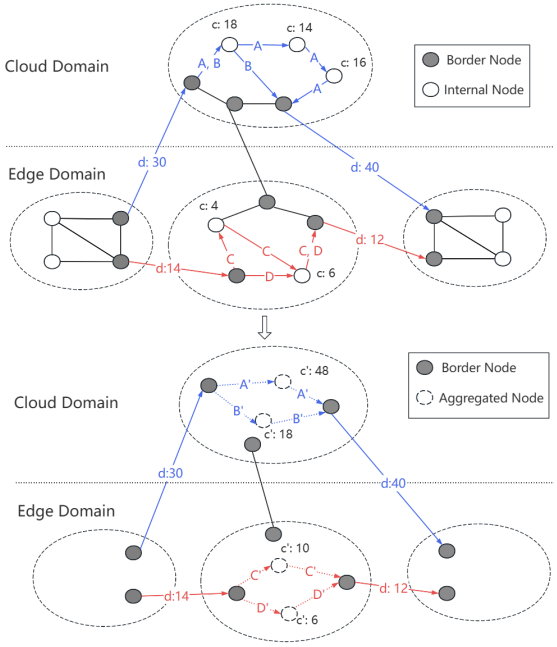


Fig. 1. Path Aggregation Example for Two Slices

Topology Aggregation (TA) enables domains to generate aggregated topologies that contain abstracted information regarding the availability of computation and network resources. For multi-domain slice provisioning, adopting TA can reduce the size of the domain topology representation and hide sensitive information of domains [2].

In this paper, we propose a reinforcement learning (RL)-based network slice composition framework to minimize the slice deployment cost in multi-domain networks. This framework incorporates a Graph Neural Network (GNN) to encode the aggregated network state of domains, an RL agent to generate slice composition plans for slice path construction, and an SFC partition algorithm to make service functions placement decisions. To the best of our knowledge, this paper is the first to apply a RL-based scheme for the network slice provisioning problem in a multi-domain environment with TA.

The remainder of this paper is organized as follows. Section II introduces related works on multi-domain network slice provisioning problem. We describe the system model and formulate the problem in Section III. Next, we illustrate our RL-based network slice composition framework in Section IV. In Section V, we design experiments to evaluate the effectiveness of the proposed scheme. Finally, we conclude the paper in Section VI.

II. Related Works

In [2], the authors proposed a TA method for the deployment of SFC across multiple optical networks. This work explores the aggregation degree of nodes and links to evaluate information compression during TA, and proposes two cross-domain path provisioning algorithm. In [3], the authors proposed a reliability-aware RL-based SFC scheduling plan. The authors assume that all the SFC requests cannot be known in advance, and the SFC is placed by finding the location each VNF in order. In [4], the authors proposed two heuristic algorithms for SFC placement in large scale cloud/edge environments with cost and latency considerations. Our previous work in [5] proposed an RL based network slice provisioning framework to improve the system profit but did not consider how to partition the SFC. Our framework proposed in this paper gives a solution to this partition problem and also considers how to determine domain sequence routing.

III. System Model and Problem Formulation

A. Multi-Domain Network Model

The network environment can be represented as a graph $G = (D, L)$, in which D is a set of heterogeneous domains connected by a set of inter-domain links L. For a domain $d_i \in D$, its internal topology is denoted as a directed graph $G_i = (V_i, E_i)$, in which the nodes $V_i = V_i^C \cup V_i^B$ consist of computing nodes V_i^C and border nodes V_i^B and are connected by intra-domain links E_i. The endpoints of an inter-domain link correspond to the border nodes of two domains. Here we consider two types of domains: edge domains and cloud domains. Edge domains are assumed to reside near the service users, indicating less network delay but a higher deployment cost and fewer computation resources compared to cloud domains.

Based on the architecture proposed by ETSI [6], we identify three key stakeholders interacting in the multi-domain networks: slice users, domain orchestrators, and a slice coordinator. The slice users request network slices to serve their applications. They define the requirements and constraints for their respective slices. In each domain, a domain orchestrator manages physical network resources within the domain, generates aggregated topologies, and offers the aggregated topologies to the slice coordinator. The slice coordinator acts as a global entity to collaborate with domain orchestrators to generate slice composition plans for slice users.

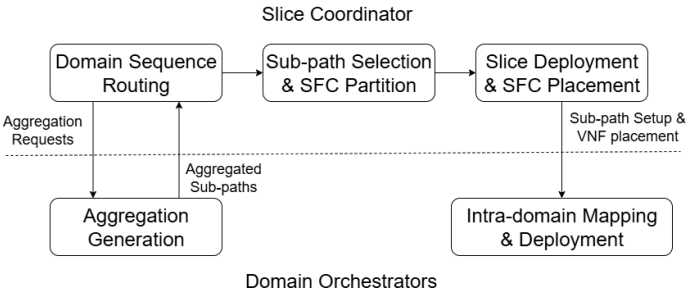


Fig. 2. Slice Composition Workflow of a Request

B. Topology Aggregation Scheme

All domains are assumed to use TA to hide their internal topologies and only expose border nodes and aggregated topologies to the outside. The aggregated topology can be formed into any structure according to specific purposes. Since the service traffic of a slice always enters and leaves at domains' border nodes (they form an "ingress-egress" pair) and uses the computing and networking resources in the domain, in this paper, we assume that all domains adopt a linear aggregation scheme for the ingress-egress pair used by the service. The aggregated linear path includes a virtual computing node between the ingress and egress nodes, and domain orchestrators may provide multiple candidate aggregated sub-paths for an ingress-egress pair so that the global coordinator can select and make an optimal E2E slice composition.

Assume that the slice coordinator inquires about aggregated sub-paths in domain d_i from u to v, $u, v \in V_i^B$. The domain orchestrator will find a set of intra-domain paths in G_i and map them as aggregated sub-paths in return. Assume there are φ aggregated paths for each ingress-egress pair, the j-th path is denoted as a virtual path $P_i^{u,v}(j) : u - v_C^{'} - v$, in which $v_C^{'}$ is the virtual computing node mapped from the computing nodes on the physical path. The performance features of this aggregation path are described as $A_{i,j}^{u,v}(R_{cmp}^{'}, R_{bw}^{'}, \tau)$, where $R_{cmp}^{'}$ is the cumulative computation capacity of computing nodes along the path, and $R_{bw}^{'}$ is the residual bandwidth capacity, and τ is the propagation delay of this path. An illustrative example is shown in Fig. 1, in which the cloud and edge domain generate aggregated sub-paths $A^{'}$ and $B^{'}$ for the slice in blue and $C^{'}$ and $D^{'}$ for the slice in red. The blue slice has a longer inter-domain delay (marked by d) than the red slice, but its aggregated sub-paths owns higher computation capacity (marked by $c^{'}$).

C. Network Slice Composition Scheme

The network service request from slice users is assumed to originate and terminate at specific domain border nodes and will be served by an SFC. The i-th request is denoted as a tuple:

$$\left\{ src, dst, b_i, \tau_i, \xi_i, \left[f_i^1, f_i^2, \ldots, f_i^{m_i} \right] \right\}. \tag{1}$$

The elements in the tuple correspond to the source and destination border nodes, required bandwidth, E2E propagation

delay bound, service duration time, and an SFC formed by m_i ordered VNFs. The r-th function in the chain requires σ_i^r units of computing resource. To serve such request, a slice coordinator collaborates with domain orchestrators to complete the network slice composition process, which includes the following steps:

Step 1: The slice coordinator specifies a sequence of domains through which the service traffic will traverse, and also determines ingress-egress pair for each domain and the set of inter-domain links to be used. **Step 2:** The slice coordinator collects aggregation sub-path sets from domain orchestrators along the domain sequence and selects one sub-path from the candidate sub-path set of each domain. An End-to-end (E2E) slice path is then constructed across a sequence of domains. **Step 3:** The VNFs of this SFC are expected to be deployed along the slice path obtained. The slice coordinator needs to find a partition plan to divide this SFC into several sub-chains, and map sub-chains to domains. **Step 4:** Given the aggregation sub-path selection and sub-chain assignment, each domain orchestrator maps the VNFs in its assigned sub-chain to the computing nodes using its own policy. The entire workflow of slice composition is illustrated in Fig. 2.

D. Problem Formulation

Given a set of slice service requests S, for the i -th request $s_i \in S$, the slice coordinator needs to make a batch of composition decisions. Suppose that the selected domain sequence consists of n_i domains and the ingress-egress pairs used by these domains are denoted as:

$$D_i = \left[d_i^1, d_i^2, \ldots, d_i^{n_i}\right] \tag{2}$$

$$\Delta_i = \left[(u_i^1, v_i^1), (u_i^2, v_i^2), \ldots, (u_i^{n_i}, v_i^{n_i})\right] \tag{3}$$

The decision variables used to select aggregation sub-paths in each domain is:

$$X^k_{(u_i^j, v_i^j)} = \begin{cases} 1, & \text{if } k-th \text{ sub-path is selected in } d_i^j \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

The SFC partition decision can be decomposed as a set of VNF placement decisions denoted as:

$$Y^r_{d_i^j} = \begin{cases} 1, & \text{if } f_i^r \text{ is mapped to } d_i^j \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

The slice deployment cost of s_i consisting of bandwidth cost and computation cost is denoted as follows:

$$C_i^{BW} = b_i \xi_i \left[\sum_{j=1}^{n_i} \sum_{k=1}^{\varphi} X^k_{(u_i^j, v_i^j)} \lambda^{p_k}_{(u_i^j, v_i^j)} + \sum_{j=1}^{n_i - 1} \lambda^L_{(v_i^j, u_i^{j+1})} \right] \tag{6}$$

$$C_i^{CMP} = \xi_i \sum_{j=1}^{n_i} \sum_{r=1}^{m_i} Y^r_{d_i^j} \sigma_i^r \theta^{cmp}_{d_i^j} \tag{7}$$

Here in (6), $\lambda^L_{(v_i^j, u_i^{j+1})}$ and $\lambda^{p_k}_{(u_i^j, v_i^j)}$ are the unit bandwidth cost of inter-domain links and aggregated sub-paths. In (7), $\theta^{cmp}_{d_i^j}$, is the unit computation cost in d_i^j. If a slice deployment violated the bandwidth or computation constraint, extra backup network or computation resources are activated to complete the deployment, and the corresponding penalties are added to the reward of this deployment. Then the cost of serving a slice request s_i is defined as:

$$Cost_i = C_i^{CMP} + C_i^{BW} + \xi_i \left[|f_e| \, \eta^{cmp} + |b_e| \, \eta^{bw} \right]. \tag{8}$$

Here $|f_e|$ and $|b_e|$ are the amount of computing resources and bandwidth that violate the constraints, η^{cmp} and η^{bw} are the corresponding penalty factors. The objective of this problem is to minimize the total deployment cost of all slice requests.

$$\min \sum_{i=1}^{|S|} Cost_i \tag{9}$$

Such slice composition plans are required to meet bandwidth, computation capacity, and E2E delay constraints. The bandwidth constraint requires the network traffic of requests loaded on inter-domain links and aggregated sub-paths of domains cannot exceed their residual capacity. The computation capacity constraint requires the computation resource requirement of sub-chains cannot exceed the aggregation path's virtual computing node's residual capacity. The E2E delay constraint requires the propagation delay of slice path constructed cannot exceed the request's delay bound τ_i.

IV. Proposed Framework

A. Framework Overview

We propose a RL based framework to solve the multi-domain network slice composition problem and develop concrete implementations of the components used in the framework. An RL paradigm must specify the state representation of the environment, the design of decision agent, and the design of reward function.

In each iteration of processing a slice request, the latest aggregated topologies from domain orchestrators are used to apply a heuristic routing strategy to find a set of candidate domain sequences from the source to the destination. These sequences, along with the current state of the multi-domain network and the characteristics of the slice request, are encoded to form the environment state representation. The decision agent employs the Deep Q Network (DQN) algorithm [7], takes the state representation as input, and generates a batch of decisions to complete the slice composition task. The E2E slice is deployed on involved domains, for which the current state of multi-domain networks are updated subsequently. Finally, a negative reward is generated as the deployment cost of this slice and is sent to the agent for learning.

B. Domain Sequence Representation

Before the routing process, each domain is asked to provide all the linear aggregation sub-paths between its border nodes following the scheme specified in section III-B. Then an auxiliary topology $G^{'}$ is generated by connecting all these aggregated topologies with inter-domain links. This global version topology is then used for domain sequence representation. The slice coordinator adopts the decision process of BGP and generates a set of candidate domain sequences. In Fig. 3, three global paths are found and denoted as the candidate domain sequences. The aggregated sub-paths within each domain (e.g., the sub-paths between border nodes a1, a2 and a3 in domain A) are omitted due to space limit. The values of border nodes and virtual nodes of aggregated sub-paths used by these domain sequences are set as 1 on $G^{'}$, those unused nodes are set as 0. This graph constructed by a one-hot encoding scheme is referred to as the Domain Sequence Graph (DSG), which is further encoded as described in Section IV-C.

C. Environment State Representation

The observations of multi-domain networks adopted here include two graphs based on $G^{'}$: Residual Computation Graph (RCG) and Residual Bandwidth Graph (RBG), to reflect the residual computation and bandwidth capacities collected from domains' aggregation sub-paths. These two graphs and DSG belong to topological data structures. The encoding approach that naively concatenates all the numerical values of elements in the topology into a vector might be sensitive to the permutation of nodes and edges. To ensure that the resulting vector is more robust to different permutations, we employ three Graph Neural Networks (GNNs) to process the information from the residual graphs. We utilize NN4G [8] as the online GNN encoder.

Since NN4G focuses on learning node representation, the bandwidth information affiliated with links cannot be processed directly and needs to be transformed as node features. Besides, we expect RBG, RCG and DSG to share the same topological representation. Here we add a virtual node $v_l^{'}$ on each inter-domain link $l \in L$ on $G^{'}$ to achieve such homogeneous encoding scheme. In RBG, the residual bandwidth values of inter-domain link and aggregated sub-path are stored in $v_l^{'}$ and $v_C^{'}$ as node features so that the bandwidth information can be encoded by NN4G. In RCG, residual computation capacities are stored in $v_C^{'}$ of aggregated sub-paths. To ensure that the values on each node have a similar scale, we adopt the ratio of remaining resources to total resources as the value assigned to each node. For instance, if the total resource capacity is 10 units, the current remaining resources amount to 7 units, then the value assigned to the node would be $7/10 = 0.7$. In DSG, with previous encoding, if a candidate domain sequence uses an inter-domain link l, the value of $v_l^{'}$ is set to 1 as well. Consequently, all node values in these graphs fall within the range of 0 to 1. During training, the gradient of the loss function, representing the difference in Q values, is back-propagated to the GNN and the model parameters are updated based on this gradient.

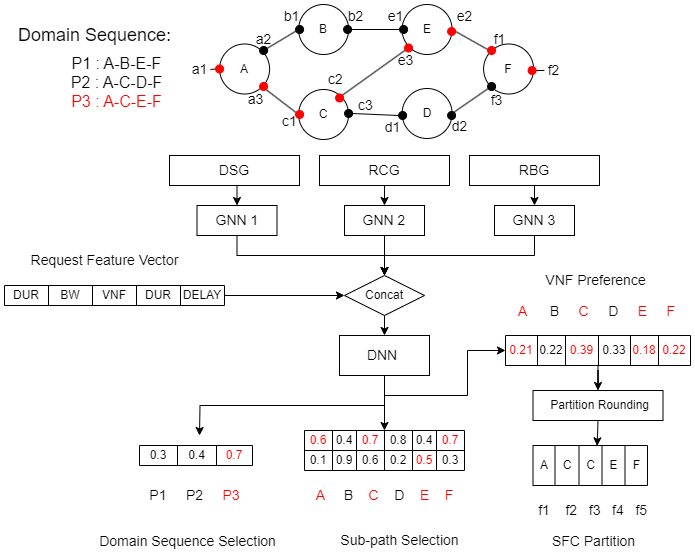


Fig. 3. RL-based Slice Composition Framework

The remaining features of the slice request are also encoded as a feature vector. Each scalar value of the request, such as bandwidth and duration time, is encoded by a single element. For the CPU requirement of VNFs, we use a vector of size m_i and mark each element with the number of CPU cores required by each VNF. By concatenating this vector with the output from NN4G, we obtain an observation vector as the input of the slice composition agent as shown in Fig. 3.

D. Agent Design

In Q-learning, the state s and action a serve as input, and a reward is obtained as output. The goal is to find the optimal policy π to obtain the optimal Q-value function $Q^*(s,a) = \max_\pi \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a, \pi\right]$, where the future rewards are discounted by γ per time step. After each action, a state transition experience $e_t = (s_t, a_t, r_t, s_{t+1})$ can be obtained, and the Q-value can be updated as $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(y_t - Q(s_t,a_t))$, in which $y_t = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$, and α is the learning rate. y_t is the target and should be equal to $Q(s_t,a_t)$ if the learning process converges. DQN does not calculate Q-values explicitly, but uses a DNN to approximate the optimum Q-values: $Q^*(s,a) \approx Q(s,a;\theta)$. θ is the weight parameter function of the DNN. In our framework, the agent takes the encoded state representation as input and outputs a composite action vector to specify a set of decisions for slice composition. The slice is then deployed along the domains according to the action and receives a reward back to the agent to evaluate its Q-value.

1) Action Space Design: The output of DNN is a composite vector, which can be decomposed into three decision vectors corresponding to domain sequence selection, aggregated sub-path selection, and VNF preference. We use a set of action vectors illustrated in Fig. 3 to explain the slice composition

decisions made for a multi-domain networks consisting of 6 domains.

Assume that there are three domain sequences found as candidates. The domain sequence selection vector contains three elements corresponding to the three domain sequences, P1, P2, and P3. The element with maximal value will be the domain sequence selection result. In Fig. 3, domains A, C, E and F are selected as the domain sequence, and the aggregation ingress-egress pairs used in these domains are also determined. Here, the aggregation pairs used in domains are (a1, a3) for A, (c1, c2) for C, (e3, e2) for E, and (f1, f2) for F.

Assume there are 2 aggregated sub-paths for each ingress-egess pair. Then the dimension for the sub-path selection vector is defined as $6 \times 2 = 12$ since each of the six domains has two elements to indicate which sub-path to use for its current ingress-egress pair if this domain is in the selected domain sequence. Similarly, the sub-path corresponding to the element with maximal value will be the sub-path selection result. The numbers marked in red in the second vector indicate that the first aggregated sub-path is selected for the ingress-egress pairs in domains A, C and F , and the second aggregated sub-path is selected for the ingress-egress pair of domain E. The E2E slice path is then established based on these two vectors.

The elements in the VNF preference vector give estimations of each domain's preferred deployment proportion of the total workload of SFC. For example, assume the SFC consists of 5 VNFs and each requires 2 units of computing resources (the total workload here is 10 units). This framework first collects the element values of domain A, C, E and F and then calls a softmax function to convert these values into normalized proportion values marked in red as shown in Fig. 3. A partition rounding function is called to find a VNF placement decisions that closely matches the preference ratio. In this case, the 1st VNF is deployed in domain A, the 2nd and 3rd VNFs are deployed in domain C, and the 4th and 5th VNF are deployed in domains E and F, respectively. In this way the VNFs of SFC are partitioned and deployed along the domains.

2) Reward Function Design: Since the objective is to minimize total slice deployment cost, we define the reward received after the deployment of each slice as the negative of the cost, which is denoted as $r_i = -Cost_i$.

V. Experiment and Evaluation

A. Experiment Settings

The multi-domain network topology used in the simulation contains two cloud domains and six edge domains, as shown in Fig. 4. An edge domain is equipped with 6 computation nodes, each containing 10 units of computation resources. A cloud domain is equipped with 12 computation nodes, each containing 20 units of computation resources. The bandwidth capacities are initialized by random selection, with intra-domain links drawn from (10 GB/s, 15 GB/s, 20 GB/s) and inter-domain links from (20 GB/s, 50 GB/s, 100 GB/s). The multi-domain simulation environment generated based on such configuration operates 400 logical time slots. The slice service

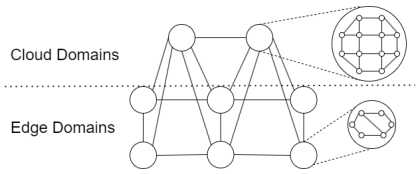


Fig. 4. Experiment Topology

requests are generated following a Poisson arrival process with parameter λ, and the service holding time follows an exponential distribution with parameter μ. For the remaining features in the request tuple, the source and destination are randomly selected from edge domains' border nodes; the bandwidth requirement is randomly selected from (200 MB/s, 500 MB/s, 1GB/s); the computation requirement of each VNF is randomly generated within the range 1 to 4 units. We set the RL algorithm to run for 500 episodes. In each simulation episode, the RL agent generates slice composition decisions for all requests arriving in the logical simulation time.

B. Baseline Algorithms

The first baseline is implemented based on a multi-domain heuristic slice construction approach proposed in [2], in which the domain sequence is greedily selected with the fewest domains traversed, and the complete slice path is constructed by choosing aggregated sub-paths with the minimum end-to-end delay. We denote this approach as "Greedy". To evaluate the effectiveness of GNN encoders, we implement a second baseline with a RL based slice composition agent without GNN, in which the domain sequence is represented as a vector using one-hot encoding. The residual numbers in RCG and RBG are also naively concatenated as the state vectors. This implementation is denoted as "Raw+RL". With TA, our proposed framework is implemented and denoted as "GNN+RL". Since the adoption of TA provides limited intra-domain information to the slice coordinator, our proposed framework is further evaluated without TA to investigate its upper-bound performance in slice composition. This implementation serves as the third baseline and is denoted as "GNN+RL w/o TA", in which all domains disclose their link and node states to construct a full observable global topology as input to the GNN encoder. Correspondingly, the action space of the RL agent is expanded to make concrete slice composition decisions over intra-domain links and nodes.

C. Performance Results

We compare our proposed framework "GNN+RL" with three baselines in terms of total deployment cost, and the result is shown in Fig. 5. The performance of "GNN+RL" converges when the training passes around 100 episodes, and the total deployment cost is lower than the heuristic scheme "Greedy" by at least 17%. "Raw+RL" yields 11% lower deployment cost than "Greedy", and its performance has a higher variance than "GNN+RL", which indicates that the encoded state representation of GNN exhibits better stability in training. "GNN+RL w/o TA" achieves 2% lower cost than "GNN+RL", indicating

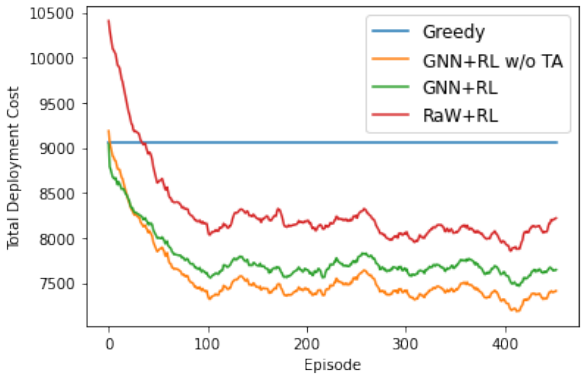
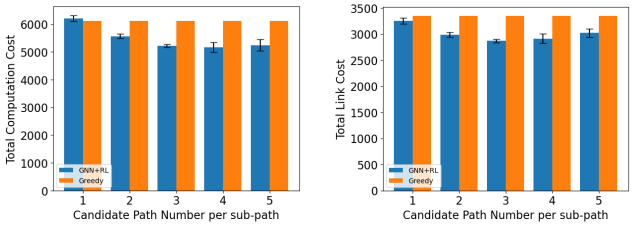


Fig. 5. Performance Comparison



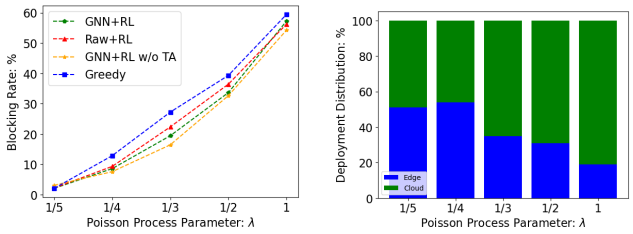
(a) Computation Cost (b) Link Cost

Fig. 6. Influence of Candidate Sub-path Number

that more efficient slice composition decisions can be made by learning from fully exposed topology without aggregation.

In order to explore how many aggregated sub-paths should be generated for an ingress-egress pair, we increase the candidate sub-path number from 1 to 5 to compare their performance with the greedy approach. The result is depicted in Fig. 6. As the candidate sub-paths increase to 3, the total deployment cost of proposed framework decreases since more candidate sub-paths give the agent more decision space to generate better sub-path combinations and partition plan. However, when the candidate number increases from 3 to 5, the total deployment cost no longer decreases and yields higher variance since the too large decision space may be overly sensitive to noise.

Extensive evaluations are conducted with different slice request arrival intensities λ. The mean blocking rate over 10 simulation runs, with 200 requests each, is illustrated in Fig. 7(a). As λ increases, the proposed "GNN+RL" achieves a lower blocking rate than "Raw+RL" and "Greedy", demonstrating the more efficient composition performance under TA. Meanwhile "GNN+RL w/o TA" further outperforms "GNN+RL" since it can observe complete intra-domain information and can specify more detailed VNF placement within domains. For the accepted slice requests in the "GNN+RL" implementation, the placement distribution of VNFs between edge and cloud domains is illustrated in Fig. 7(b). When λ increases from 1/5 to 1/4, the proportion of placements in the cloud domain decreases slightly because the edge domains still have sufficient computational capacity. In this case, the



(a) Blocking Rate (b) VNF Placement Distribution

Fig. 7. Influence of Arrival Intensity

agent tends to place VNFs in edge domains to better satisfy delay-sensitive slice requests. However, as λ increases beyond 1/4 up to 1, the resources in the edge domains are more rapidly exhausted, leading to a higher proportion of VNFs being placed in cloud domains.

VI. Conclusion

In this paper, we proposed a linear topology aggregation mechanism, upon which a multi-domain network slice composition problem is formulated. To solve this problem, we developed an RL based framework in which the agent leverages a GNN-based encoder to learn an aggregated representation of domain states. Based on this representation, the agent makes a series of slice composition decisions, including domain sequence routing, aggregated sub-path selection, and SFC partitioning. Experimental results demonstrate that our framework achieves a 17% reduction in total slice deployment cost compared to a greedy baseline.

VII. Acknowledgment

This work was supported in part by the National Science Foundation under Grant No. CNS-2008856.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE communications magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] B. Yan, Y. Zhao, X. Yu, Y. Li, S. Rahman, Y. He, X. Xin, and J. Zhang, "Service function path provisioning with topology aggregation in multi-domain optical networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2755–2767, 2020.

[3] J. Jia, L. Yang, and J. Cao, "Reliability-aware dynamic service chain scheduling in 5g networks based on reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2021.

[4] M. A. Khoshkholghi, M. Gokan Khan, K. Alizadeh Noghani, J. Taheri, D. Bhamare, A. Kassler, Z. Xiang, S. Deng, and X. Yang, "Service function chain placement for joint cost and latency optimization," *Mobile Networks and Applications*, vol. 25, pp. 2191–2205, 2020.

[5] Z. Wu, G. Ishigaki, R. Gour, C. Li, F. Mi, S. Talluri, and J. P. Jue, "Reinforcement learning-based multi-domain network slice provisioning," in *ICC 2023 - IEEE International Conference on Communications*, pp. 1899–1904, 2023.

[6] ETSI, "Next generation protocols (NGP); large-scale deterministic network," *E2E Network Slicing Reference Framework and Information Model*, 2018.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.