Migration Aware Virtual Network Embedding in Software Define Networks

Arezoo Jahani[1]

[1]*Faculty of Electrical and Computer Engineering, Sahand University of Technology, Tabriz, Iran*
a.jahani@sut.ac.ir

Abstract

The development of Software Define Networks (SDN) networks and the of Network Function Virtualization (NFV) have provided the sharing of resources for cloud service providers. Developing effective virtual network embedding (VNE) algorithms for an SDN network is crucial to improve resource utilization. However, after allocating resources to a virtual network request, most existing VNE algorithms allocate the same resources to that request until the end of execution, which causes this problem in most cases; due to the existence of disconnection in the physical network graph, even with empty resources, it is not possible to map a new request. The proposed MaVNE method in this article models the VNE as a MILP problem and while calculating the migration cost of virtual networks, it tries to address the mentioned problem. The results of evaluation and comparison of the proposed method with basic methods show the power of the proposed method in increasing the acceptance ratio of virtual networks. But the cost of migration is added to the migrated networks. Therefore, by increasing the cost, which is the cost of immigration, the acceptance ratio can be increased.

Keywords: *Network Embedding, Network virtualization, Linear problem, Migration, Software Define Network.*

1 Introduction

The rapid advancement of cloud computing services and virtualization technologies has transformed how network resources are managed and utilized. Among these technologies, Software-Defined Networking (SDN) and Network Function Virtualization (NFV) have emerged as key enablers, allowing for more flexible and efficient network management which make it possible to easily share resources between users using cloud resources. SDN decouples the control and data planes, providing centralized control over network resources, while NFV facilitates the virtualization of network functions, enabling dynamic and scalable deployment of services. Together, these technologies

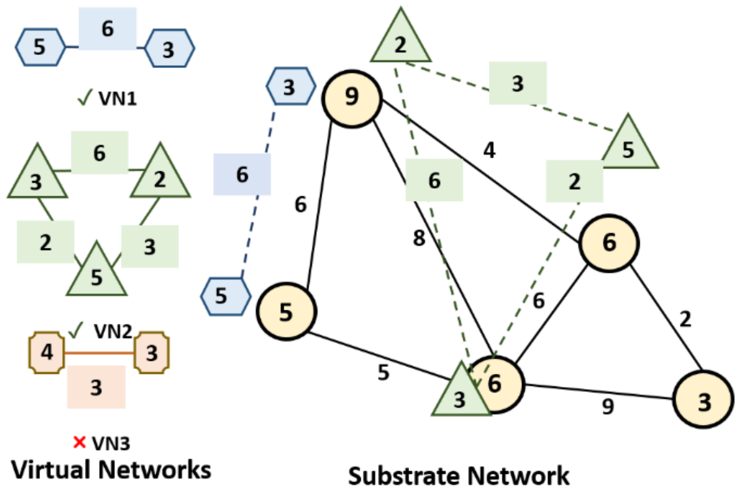

Figure 1: An example of two virtual networks embedding

have paved the way for the efficient sharing of network resources among multiple cloud service providers, giving rise to the concept of Virtual Network Embedding (VNE) [1].

VNE is a fundamental process in SDN-enabled networks that involves mapping virtual networks (VNs) onto a physical substrate network. The goal is to allocate the physical resources such as bandwidth, CPU, and memory to meet the demands of multiple VNs while optimizing resource utilization across the entire network. Effective VNE algorithms are critical for enhancing the overall efficiency and performance of SDN-based networks [2]. These algorithms must address the challenges of resource allocation in a dynamic and often unpredictable network environment, where demand can fluctuate and network topologies may change. An example of VNE is demonstrated in Fig. 1.

As shown in Fig. 1, there is a substrate network with five nodes whose resource capacity is written on nodes and seven links which their bandwidth capacity is written on links. Also, there are three different VNs with node requirements and link bandwidths. A VNE problem is looking to find a better place on the substrate network for VNs and the goal is increasing the acceptance ratio and revenue, and decreasing the cost. In Fig. 1, VN1 and VN2 are embedded successfully, but VN3 is not embedded, although there are enough resources, the resources are disconnected.

However, a significant limitation of most existing VNE algorithms is their static approach to resource allocation. Once resources are assigned to a VN request, they remain fixed for the duration of the request, regardless of changes in network conditions. This static allocation can lead to inefficiencies, particularly in scenarios where the physical network graph experiences disconnections or where available resources become fragmented [1, 3]. In such cases, even when there are sufficient resources available, the inability to reallocate or migrate resources dynamically can prevent the accommo-

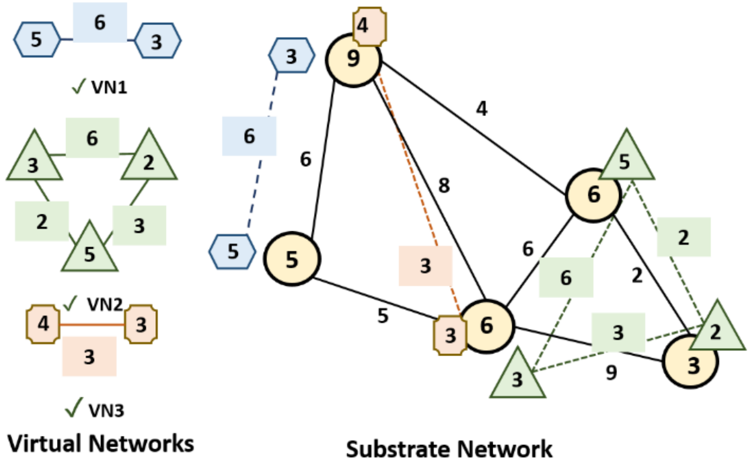

Figure 2: An example of substrate and virtual network

dation of new VN requests, thereby reducing the acceptance ratio of the network. This problem is shown in Fig. 1 which the VN3 is not mapped successfully, although there are enough resources. The problem is that the resources are disconnected and just by changing the mapping location of one of the requests, enough connected space will be created for the third request as shown in Fig. 2.

As shown in Fig. 2, imagine that VN3 is arrived after VN1 and VN2. After embedding the VN1 and VN2 like Fig. 1, there is not any enough connected resource to assign to VN3. But by changing the mapping location of one of the requests, enough connected space has been created for the third request as well.

To address these challenges, this paper proposes a Migration-Aware Virtual Network Embedding (MaVNE) approach that models the VNE problem as a Mixed Integer Linear Programming (MILP) problem. The MaVNE method introduces a dynamic resource allocation strategy by incorporating migration costs into the VNE process. By calculating the potential migration cost and considering it in the decision-making process, the MaVNE approach aims to improve the overall acceptance ratio of VN requests while managing the trade-off between resource utilization and migration overhead.

The rest of the paper is organized as follows: Section 2 provides a review of related work in the field of VNE. Section 3 details the proposed MaVNE methodology, including the formulation of the MILP problem and the migration cost model. Section 4 presents the experimental setup and results, comparing the performance of the proposed method with baseline approaches. Finally, Section 5 concludes the paper with a discussion of the findings and potential directions for future research.

2 Related Works

The VNE has attracted a lot of attention in recent years. The proposed methods for this problem can be examined from different aspects, most of which are shown in Fig. 3.

As seen in Fig. 3, VNE problems can be classified into two categories: static or dynamic. Static problems are problems in which all requests are known at the beginning of resource allocation, and resource allocation is done once, and all requests must use the same resources until the end of execution. But in dynamic mode, requests are entered into the system in order, and the number of times and the time of execution of the proposed algorithm is usually uncertain and depends on things such as the time of entering a new request [1].

The next item is the objectives of algorithms related to VNE problems, which include: cost [2], revenue, security [3], acceptance rate, and topology [4]. Most of the articles related to VNE seek to increase the revenue of the service provider, reduce the cost of the user and increase the acceptance ratio in the sense of increasing the number of received requests. Some of the latest articles also examine security and try to use physical resources that have the same security level as virtual nodes as much as possible to map virtual networks. Some articles also look for mapping based on network topology and try to use resources that lead to non-separation of the network by knowing the network topology.

Mapping of virtual networks is done in two phases: node mapping and link mapping. Node mapping is the concept of choosing suitable physical nodes for virtual nodes, which is usually done by methods such as ranking [5] or prioritizing nodes, or giving credit to nodes. Link mapping is also usually done by algorithms of finding the shortest path or commodity flow. Now, these two phases can be done in coordination [6] with each other or uncoordinated. Coordination means to think about the link mapping phase in the node mapping phase and sometimes to change the node mapping in order to achieve the goals in the next phase. This mode itself can be classified into two other cases, which are two-phase and single-phase. In the two-phase stage, the two stages of node and link mapping are performed separately and sequentially, but in the single-phase stage, the two stages are performed completely side by side.

VNE algorithms can be executed in one of two central [7] or distributed [8] modes. In centralized mode, only one node is responsible for managing the entire physical network. But in the distributed mode, the responsibility of managing the network in a distributed manner is the responsibility of several components that must communicate with each other peacefully. Also, the methods proposed in previous articles can be classified as linear programming methods, statistical methods [9], or methods based on machine learning or deep learning.

Linear programming based methods use problem modeling as a linear or non-linear programming problem and then look for a solution by searching the complete or pruned state space [10]. But statistical methods by estimating the result and machine learning

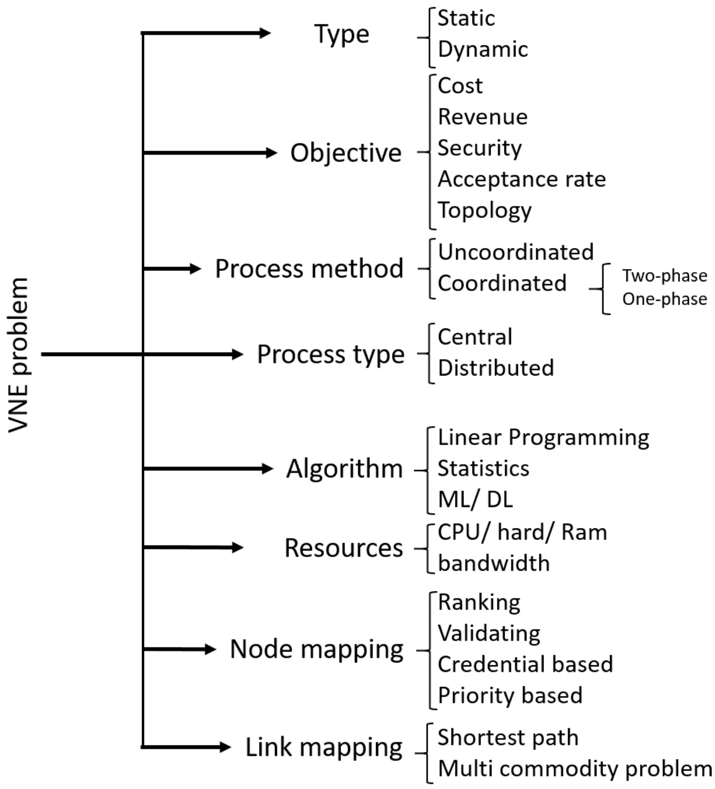

Figure 3: Classification of VNE problems

methods by learning and testing do this [11]. In node mapping, the resources that must be allocated from physical nodes to virtual nodes include processor, memory, and hard drive, and in link mapping, resources such as bandwidth are considered.

3 Proposed MaVNE method

The proposed MaVNE method is a VNE solution which model it as a Mixed-Integer Linear Programming (MILP). The main contribution of the proposed method is its migration awareness. After allocating resources to a request, the previous methods did not reclaim the resources until the end of the execution, and this problem led to the problem shown in Fig. 1 and 2, which the proposed method addressed. Therefore, the proposed method has a modeling of the VNE problem that this model is re-executed during the following three states and each time it is executed, all allocated resources are withdrawn to be re-allocated.

- A new virtual network request arrives
- Completing the arrival of an allocated virtual request

Table 1: Notation

Parameters	
$G_s = (N_s, L_s)$	Substrate network
$N_s = \{R_i \mid i \in N_s\}$	Substrate nodes with their amount of resource
$L_s = \{BW_{ij} \mid i \to j \in L_s\}$	Substrate links with their amount of bandwidth
$G_v = (N_v, L_v)$	Virtual network
$N_v = \{R_u \mid u \in N_v\}$	Virtual nodes with their amount of request
$L_v = \{BW_{uv} \mid u, v \ \in N_v, \ u \to v \ \in L_v\}$	Virtual links with their amount of required bandwidth
$R_v = (G_v, Stime_v, deadline_v)$	Virtual Request with their submitted time and deadline
i, j	Substrate nodes
u, v	Virtual nodes
$cost_{migration}$	Migration cost
M	A fix time size for extending the simulation time
Variables	
X_{uv}^{ij}	Binary variable, virtual link $u \to v$ is mapped on substrate link $i \to j$
Y_v^i	Binary variable, virtual node v is mapped on substrate node i
$delay_v^t$	Delay of virtual request v at time t
P_{uv}	The path for mapping virtual link L_{uv}

- Elapse of a predetermined fixed period of time

But we must note that with this work, a request may receive different resources during two consecutive executions, which forces migration for these types of requests.

The required notations are demonstrated in Table 1 and as demonstrated we model the substrate and virtual networks as an undirected graph with nodes and links. The nodes have resources like CPU, Ram, and hard. The links have resource like bandwidth.

The proposed MILP model for VNE is shown in Equation P1 which is a minimization problem. The objective function has fine different sections which are combined and are; (I) node mapping cost, (II) link mapping cost, (III) migration cost, if the location of nodes are changed, (IV) migration cost, if the location of links are changed, and (V) delay of virtual requests. In fact, the model minimize the embedding cost which is based on the amount of resources are used for each mapping (both of the nodes and links), the migration cost which is based on the happened migration for both of the nodes and links, and the delay for all available requests.

The constraints are shown in Equations (p1a) to (p1g). In fact, constraint (p1a) demonstrates that the amount of bandwidth of physical links allocated to virtual links should be sufficient. (p1b) demonstrates that the amount of resources required by virtual nodes should not be more than the resources available in physical nodes. Constraint (p1c) demonstrates each virtual link should be mapped on a physical path whose size is equal to the size of the path for which the virtual link is selected. Constraint (p1d) demonstrates each virtual node should mapped on only one physical node. Constraint (p1e) demonstrates that the delay of each virtual request should computed with comparing the simulation time (t) with their deadline time. Constraints (p1f)-(p1h) manage the range of possible answers.

$$\min \sum_{i,j\in G_s}\sum_{u,v\ \in G_v} X_{uv}^{ij} * \ BW\left(L_{uv}\right) \tag{p1}$$

$$+\sum_{i\in G_s}\sum_{v\in G_v} R_v * Y_v^i$$

$$+\sum_{i,j\in G_s}\sum_{u,v\in G_v}\sum_{t\in time} cost_{migration}\left(\left(X_{uv}^{ij}\right)_t - \left(X_{uv}^{ij}\right)_{t-1}\right)$$

$$+\sum_{i\in G_s}\sum_{v\in G_v} cost_{migration}\left(\left(Y_v^i\right)_t - \left(Y_v^i\right)_{t-1}\right) + \sum_{R_v}\sum_{t\in time} delay_v$$

$s.t:$

$$\sum_{i,j\in G_s} X_{uv}^{ij} * BW\left(L_{uv}\right) \le BW(L_{ij}) \qquad \forall u,v\in G_v \tag{p1a}$$

$$\sum_{i\in G_s} Y_v^i * R_v \le R_i \qquad \forall v\in G_v \tag{p1b}$$

$$\sum_{i,j\in G_s} X_{uv}^{ij} \le |P_{u,v}| \qquad \forall i,j\in G_s, \forall u,v\in G_v \tag{p1c}$$

$$\sum_{i\in G_s} Y_v^i \le 1 \qquad \forall i\in G_s, \forall v\in G_v \tag{p1d}$$

$$delay_v^t \ge \sum_{v\in N_s} Y_v^i * (t - delay_v) \qquad \forall t\in time, \forall v\in G_v \tag{p1e}$$

$$X_{uv}^{ij} \in \{0,1\} \tag{p1f}$$

$$Y_v^i \in \{0,1\} \tag{p1g}$$

$$delay_v^t \ge 0\ ,\ \ P_{uv} \ge 0 \tag{p1h}$$

As shown in problem (p1), the simulation time is the time which passed by one of these events (receiving a new request, completing one of the accepted requests, or passing a fix time size) and its initial value is zero. For better understanding focus on Fig. 4.

As shown in Fig. 4, imagine that there are four requested networks which three of them arrived at time zero. So, the first simulation time (sim_time0) is equal to zero. In order to extend the simulation time, there is three rules: (1) a request is completed. (2) a new request is arrived. (3) a fix time is elapsed.

With extending the simulation time, the MILP model is executed again. But for each execution of the proposed MILP model, all of the assigned resources should be released and all of the not completed requests should be update their remained time for using the resources. This is the point that can address the problem shown in Fig. 1, and Fig. 2.

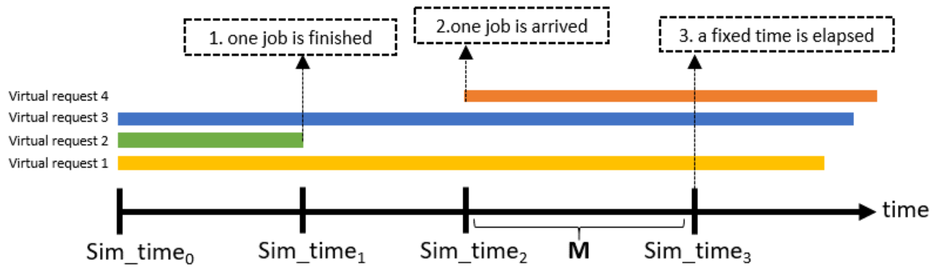

Figure 4: Three different events for expanding the simulation time

Table 2: Evaluation parameters

Substrate network	
Node numbers, link numbers	20, 54
Node capacity	[20, 70], uniform distribution
Link bandwidth	[10, 50], uniform distribution
Virtual network	
Node numbers, link numbers	[2,4], [3,7]
Node capacity	[2, 10], uniform distribution
Link bandwidth	[1, 15], uniform distribution
Arrival time	Poisson distribution
deadline	random
Fix time	20 second
Simulation	
Language, framework	Python, Pyomo
Mipgap = 0.5	A Pyomo parameter for search tree pruning

4 Evaluation of MaVNE

In order to evaluate the proposed MaVNE method, we used a random topology for substrate and virtual networks which are formed by NSG2.1 (graphical GT-ITM) [13]. The used parameters are shown in Table 2.

As shown in Table 2, we imaged that there is a substrate network with 20 nodes and 54 links. The nodes' capacity obeys a uniform distribution, between 20 and 70 and the link bandwidth also obeys the uniform distribution which is between 10 and 50. The virtual network has minimum 2 nodes and maximum 4 nodes. Also have minimum 3 and maximum 7 links. The requests arrival time obeys the Poisson distribution with random number for deadline. We implemented the proposed MaVNE and the previous paper EE-CTA in python, Pyomo and evaluated it on a Windows based system with 16G Ram, and a CPU corei7, 1.80 GHz, 11 generation.

4.1 Simulation results

We compared the proposed MaVNE with EE-CTA [1] which was a topology-aware VNE method and used a decomposition method for slicing the substrate network for better

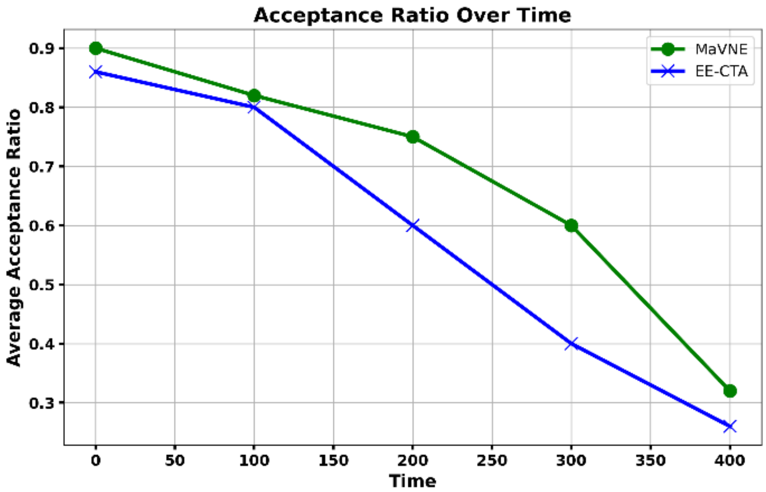

Figure 5: Average acceptance ratio over time

mapping to avoiding the problem of mentioned in Fig. 1 and Fig. 2. So, the EE-CTA was the best method for comparing with our proposed method. The acceptance ratio and the cost for different number of VNs are demonstrated respectively in Fig. 5 and Fig. 6.

As shown in Fig. 5, the average acceptance ratio decreases over time, which is due to busy resources due to an increase in the number of requests. Requests are entered into the system with Poisson distribution, and with the arrival of each new request or the completion of one of the previous requests or the passage of a fixed period of time, the allocation of resources for all requests starts again from the beginning. Also, for each reassignment, the time required for the tasks that had previously received resources but have not yet been completed is recalculated. According to Fig. 5, the proposed MaVNE method has a higher acceptance rate compared to the previous EE-CTA method, and this increase in acceptance rate is due to the ability of the proposed method to reallocate resources for all new requests and requests that have not yet been completed. We have assumed that we know the time required to execute the requests. While in the real environment, it is necessary to calculate the required time of requests usually by methods such as estimation or prediction.

As shown in Fig. 6, the cost spent on mapping requests increases with time. We have calculated this cost in two ways. One is when we have measured the migration cost of the proposed method and the other is when we have ignored the migration cost of the proposed method. The first case is shown in Fig. 6 and the second case is shown in Fig. 7.

When we included the migration cost for the requests that used different resources during different time periods, the average cost in the proposed method of MaVNE was higher than the previous EE-CTA method. However, it is necessary to bear the cost

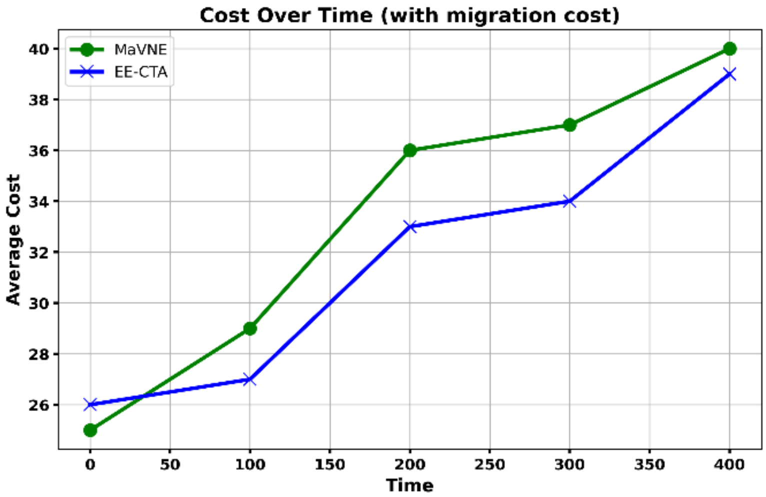

Figure 6: Average cost over time (with migration cost)

of immigration in order to increase the acceptance rate. Also, to prove the issue once again in Fig. 7, we omitted the cost of migration and the results show that in this case, the cost was significantly reduced compared to the previous method, but it is natural that in the real environment, the cost of migration cannot be omitted. And the bottom line is that in order to increase the acceptance rate, the cost of immigration must be borne. Note that the cost of migration does not cause a problem in the previous EE-CTA method. Because the previous method does not allow requests to migrate, and all requests from the first time they receive resources, must use the same resources until the end of execution and are not allowed to change resources.

As the last evaluation test, we compare the execution time of the proposed MaVNE and EE-CTA and the results are shown in Fig. 8.

As demonstrated in Fig. 8, the average execution time in the proposed MaVNE method is as close as the previous EE-CTA method. Although the EE-CTA sometimes has less execution time, it's because of not adding the executed requests to the allocation process again. However, the proposed MaVNE for each running the VNE problem, releases all the resources and assigns again all of the resources from the beginning.

5 Conclusion and future work

In order to solve the problem of not being able to allocate resources when there are sufficient and unrelated resources, MaVNE method was proposed. The proposed method can update the simulation time in the following three cases: when a running request ends. When a new request enters the system. When a fixed predetermined period of time elapses. With these explanations, the proposed MaVNE method can first retrieve all the available resources and then map all the resources from the beginning every time

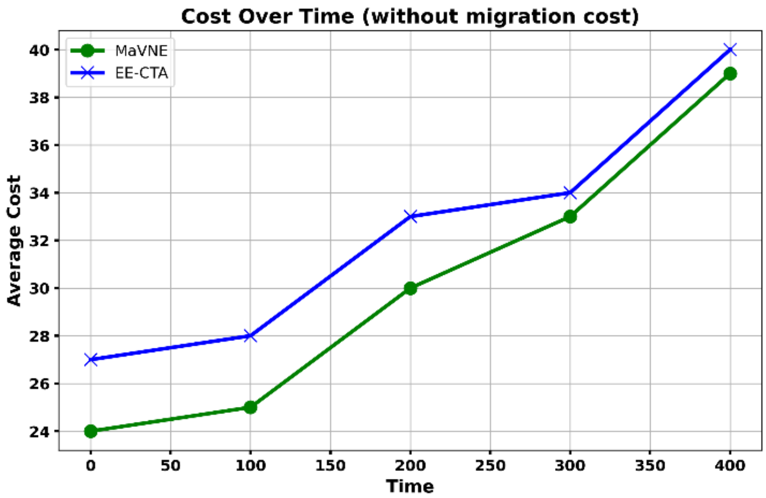

Figure 7: Average cost over time (without migration cost)

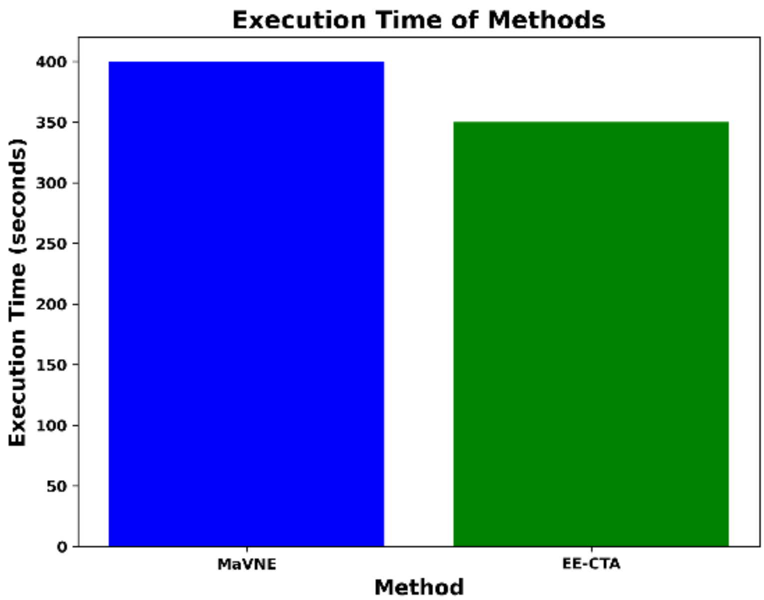

Figure 8: Average execution time

the resource allocation algorithm is re-executed. This will increase the acceptance rate. However, the cost of mapping increases due to the need to migrate requests to which the assigned resource has changed. Therefore, in order to increase the acceptance rate, it is necessary to bear the cost of immigration.

References

[1] Jahani, Arezoo, et al. “EE-CTA: Energy efficient, concurrent and topology-aware virtual network embedding as a multi-objective optimization problem”. Computer Standards & Interfaces, Vol. 66, 2019, pp. 103351.

[2] Nguyen, Huu Thanh, et al. “A generalized resource allocation framework in support of multi-layer virtual network embedding based on SDN”. Computer Networks 92, Vol. 2015, pp. 251-269.

[3] Wang, Yang, et al. “Minimum-cost embedding of virtual networks: An iterative decomposition approach”. Computer Networks, Vol. 234, 2023, pp. 109907.

[4] Liu, Shuhao, et al. “Towards security-aware virtual network embedding”. Computer Networks, Vol. 91, 2015, pp. 151-163.

[5] Cheng, Xiang, et al. “Virtual network embedding through topology awareness and optimization”. Computer Networks, Vol. 56, Issue. 6, 2012, pp. 1797-1813.

[6] TG, Keerthan Kumar, et al. “NORD: NOde Ranking-based efficient virtual network embedding over single Domain substrate networks”. Computer Networks, Vol. 225, 2023, pp. 109661.

[7] Duan, Zhonglei, and Ting Wang. “Towards learning-based energy-efficient online coordinated virtual network embedding framework”. Computer Networks, Vol. 239, 2024, pp. 110139.

[8] Zhang, Shengyu, and Kwan L. Yeung. “Location constrained virtual optical network embedding in space-division multiplexing elastic optical networks”. Computer Networks, Vol. 220, 2023, pp. 109475.

[9] Beck, Michael Till, et al. “Distributed and scalable embedding of virtual networks”. Journal of Network and Computer Applications Vol. 56, 2015, pp. 124-136.

[10] Dandachi, Ghina, et al. “A robust control-theory-based exploration strategy in deep reinforcement learning for virtual network embedding”. Computer Networks, Vol. 218, 2022, pp. 109366.

[11] Moreira, Cristiano L., Carlos A. Kamienski, and Reinaldo AC Bianchi. “5G and edge: A reinforcement learning approach for Virtual Network Embedding with cost optimization and improved acceptance rate”. Computer Networks, Vol. 247, 2024, pp. 110434.

[12] Lim, Hyun-Kyo, et al. “Reinforcement learning-based virtual network embedding: A comprehensive survey”. ICT Express 9.5, 2023, pp. 983-994.

[13] E.W. Zegura , K.L. Calvert, S. Bhattacharjee , How to model an internetwork, in: Proceedings of the INFOCOM’96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, Vol. 2, 1996, pp. 594–602.