Constrained dynamic virtual network embedding

Junkai He, Makhlouf Hadji, Djamal Zeghlache

Constrained Dynamic Virtual Network Embedding

1st Junkai He
Télécom SudParis
Institut Mines-Télécom
Évry, France
junkai.he@telecom-sudparis.eu

2nd Makhlouf Hadji
Technological Research Institute SystemX
Palaiseau, France
makhlouf.hadji@irt-systemx.fr

3rd Djamal Zeghlache
Télécom SudParis
Institut Mines-Télécom
Évry, France
djamal.zeghlache@telecom-sudparis.eu

***Abstract*—This paper focuses on a dynamic embedding of client-constrained heterogeneous Virtual Network (VN) requests with multiple nodes and links affinity and anti-affinity requirements. For this Virtual Network Embedding (VNE) problem, we formulate an Integer-Linear Programming (ILP)-based model that achieves joint mapping of virtual nodes and links of each VN onto the dynamically updated Substrate Network (SN). This model not only meets the clients expressed isolation constraints but also includes VN request arrivals and departures to update SN information. Numerical experiments illustrate the efficiency of the proposed methods and their ability to find optimal solutions. Performance reports provide cloud service providers with insights into additional investments in nodes and links they should make to serve clients with anti-affinity requirements.**

***Index Terms*—virtual network embedding, customized request, isolation, anti-affinity, integer-linear programming**

I. Introduction and Related works

Next-generation networks and digital infrastructures evolving to programmable and dynamically established systems rely on virtualization technologies [1] but face allocation, provisioning, and mapping challenges in nodes and links between the virtual and physical worlds. Mapping virtual nodes and links of Virtual Networks (VN) according to diverse clients requirements and constraints onto physical networks (also known as Substrate Networks, SN) is the well-known Virtual Network Embedding (VNE) problem [2]. So far, VNE solutions have partially addressed heterogeneity and differences in clients embedding requests [3]. Clients have major and significant differences in requirements, affinity, and anti-affinity constraints in nodes and links in their own VNs as well as with other clients VNs. How to provide differentiated embedding decisions to customer-specific VNs and associated services and applications remains a concern. The heterogeneity and specific preferences in modern customized VNs have to be further investigated when sharing physical infrastructures to serve multiple users.

A. VNE with Node/Link Anti-affinity

This paper addresses multiple and heterogeneous VN requests from clients that have very diverse requirements in terms of affinity and anti-affinity between VNs of their own request and tolerance or very stringent requirements with respect to other clients. Such a generalized VNE problem taking into account these types of constraints within a client VN and across multiple clients VNs has received little if no attention. Some clients are likely to require isolation and separation of their virtual nodes and links for security, trust, resilience, and performance reasons imposing constraints on co-localization on hosting nodes and separation of hosting links or end-to-end paths across clients. Clients are in general end users, consumers and tenants. Most existing papers focus on node (anti-)affinity asking for separation or co-localization of their VNs in hosting nodes. Previous works with node anti-affinity include [4]- [10] but link anti-affinity is rarely studied in the literature. Even if dedicated resources can be requested in nodes and links (or dedicated paths) from providers, there is no generalized VNE modeling framework including all these aspects by design. VNE with joint node anti-affinity and link anti-affinity are seldom addressed except for [11] and [12] that focus on SFC (Service Function Chain) embedding, which can be seen as a special case of VNE. Work of [11] proposes a set of affinity and anti-affinity constraints in a SFC concerning virtual nodes and links. Once a SFC is embedded, its corresponding physical configuration cannot be easily changed and thus becomes static and no adjustment or dynamic changes are envisaged [12]. A SFC contains the source node and (multiple) sink nodes [13] and its embedding should respect flow conservation constraints in the chain. Ref. [12] extends the work [11] by proposing a non-linear integer programming formulation and a heuristic algorithm accomplishing joint node and link mappings. Further, [11] and [12], relying on simulations rather than formal optimization, do not consider dynamic arrivals and departures of VNs and represent in non-linear form anti-affinity. By contrast, we include in our model from the start node and link anti-affinities among VN requests, map virtual nodes and virtual links dynamically, and comply with all anti-affinity constraints. Our goal is to propose a generic solution for multiple heterogeneous VN requests that can be fine-tuned to meet a simple graph embedding need as well as much more complex requests such as multiple SFC and slicing demands.

B. VNE Strategies and Solution Methods

Current state-of-the-art addresses VNE, known to be NP-hard [2], typically in a two-stage approach, embedding nodes first and then links. Some authors proposed joint nodes and links mapping using either ILP-based models that do not scale with problem size [4], [6], [14], [15] or heuristic algorithms (such as [1]) to complete in some cases the mapping in

polynomial time. Work of [16] proposes an improved VNE algorithm to embed nodes and links in a single stage with significantly reduced computational compared to ILP models. Authors of [17] rely on node degree to map nodes and links simultaneously while [18] uses compatibility graph theory to accomplish the joint embedding. In these efforts to realize simultaneous joint mapping of nodes and links, VN affinity and anti-affinity requirements expressed by end users have seldom been investigated thoroughly. To the best of our knowledge, no generalized and generic modeling frameworks that take into account intra and inter VN constraints from multiple user requests have been proposed in the literature. This paper aims at filling this gap by addressing heterogeneous VN requests from multiple independent users with diverse co-localization, separation and isolation requirements.

C. Motivations and Contributions

The motivations and contributions of this study include: i) taking into account heterogeneous VN requests from clients expressing various affinity and anti-affinity constraints in their own requests and requirements with respect to other clients and hosting infrastructures (an aspect seldom addressed in the current literature) and ii) propose a solution method to achieve joint embedding of nodes and links while respecting diverse and multiple clients constraints and requirements. Our work consequently:

1) derives a generic dynamic VNE optimization model that takes into account both affinity and anti-affinity requirements on nodes and links. These requirements apply not only within a client's VN request, but also among VN requests from other clients.
2) formulates an ILP model making optimal embedding decisions for each VN request and including node co-localization/separation constraints, node anti-affinity constraints and link anti-affinity constraints.
3) considers the arrivals and departures of all received VNs in the ILP-based model with dynamic updates of the hosting SN topology during the performance assessment.

This paper first describes the constrained dynamic VNE problem in Section II. Section III presents the proposed ILP-based model for the addreseed problem. Section IV reports the results of numerical tests and analyzes the outcomes to measure algorithm effectiveness. Section V draws conclusions based on our entire study and suggests possible directions for future research.

II. Problem Description

We first introduce the SN topology and client-constrained VN requests representation and then present our constrained dynamic VNE problem and model.

A. Network Topology and Attributes

The generic SN is modeled as an undirected graph $G^{\mathcal{S}} = (N^{\mathcal{S}}, L^{\mathcal{S}}, CPU^{\mathcal{S}}, BW^{\mathcal{S}})$ where $N^{\mathcal{S}}$ and $L^{\mathcal{S}}$ represent respectively the sets of physical nodes and physical links. Parameters $CPU^{\mathcal{S}}$ and $BW^{\mathcal{S}}$ denote the available computing power (CPU) of nodes and bandwidth of links, seen as attributes and resources. Note that some nodes may be connected via direct links but for sure nodes are always interconnected at least through an end-to-end path [4]. Heterogeneous VN requests can be represented as an undirected graph $G^{\mathcal{V}} = (N^{\mathcal{V}}, L^{\mathcal{V}}, CPU^{\mathcal{V}}, BW^{\mathcal{V}}, \alpha, \beta)$, where $N^{\mathcal{V}}$ and $L^{\mathcal{V}}$ represent respectively the sets of virtual nodes and virtual links. The required CPU on nodes and bandwidth on links are represented by $CPU^{\mathcal{V}}$ and $BW^{\mathcal{V}}$. As VN requests may have different structures, attributes and requirements, we use α to represent node co-localization or node separation requirements within a VN. Node co-localization ensures that two nodes within a VN request are mapped to the same physical node, while node separation guarantees their hosting on different physical nodes. The notation β indicates node anti-affinity (βn) and link anti-affinity (βl) with other clients' VN requests. If nodes and links within a VN request have anti-affinity requirements, they must be isolated from other clients' VN requests, without sharing physical nodes or links within the underlying SN. Fig. 1 depicts an example of two established heterogeneous VN requests and indicates that in VN request n-1, the nodes with 4 and 8 CPUs should be co-located on a physical node, whereas in VN request n, the nodes with 5 and 2 CPUs must be separated during the mapping process. In addition, the node with 3 CPUs and link with 5 bandwidth units in request n-1 contain anti-affinity constraints and must be isolated upon mapping to the SN. Similarly, the node with 6 CPUs and link with 7 bandwidth unit requirements in request n must also be isolated due to anti-affinity.

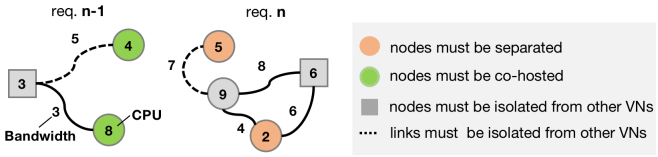

Fig. 1. Illustration of heterogeneous VN requests.

We use the arrival time and departure time to describe dynamically incoming VN requests. The arrival time refers to the time when the VN request is received for processing while the departure time corresponds to the departure of the VN and the time of the release of the used physical resources hosting the VN. Without any loss of generality, the VNs in our work are treated sequentially even though they could be treated in batch mode. Anyway, the VN requests can be single requests or combined requests in a composite graph expressing multiple mixed requests with appropriate internal affinity and anti-affinity requirements or relationships. So treating random VN with random structure and arrival times sequentially reflects all these scenarios. Hence we treat VN requests sequentially as they arrive and embed them one after the other and leave lumping requests out of the scope of this work.

B. Constrained Dynamic VNE

To successfully embed a VN request onto the SN (called an allocation step), a number of constraints must be met in

the modeling framework to obtain embedding solutions. Every virtual node has to be hosted by exactly one physical node in the SN. In addition, every virtual link has to be assigned to a direct physical link when appropriate or a path because we aim at mapping virtual nodes and links jointly onto the SN. Consequently, for every virtual link $(i,j) \in L^{\mathcal{V}}$, there must exist a path $(m,n) \in P^{\mathcal{S}}$ such that the source virtual node i and the destination virtual node j of link (i,j) are assigned to the source physical node m and destination physical node n of a selected path $(m,n) \in P^{\mathcal{S}}$, respectively. At any allocation step, if there is more than one path available between physical nodes m and n, the shortest available path is selected.

Another standard set of constraints to include in the mathematical model is the number of available resources in the SN at each allocation round and these resources must be sufficient to host the VN requests. Further, the dynamic arrival and departure times impact the SN resources and topology and this must also be taken into account in the model. The update of the SN depends on mainly three factors: i) the mapping results of received VN requests that determine the SN physical resources occupation and availability, ii) the departures of successfully mapped VN requests as they release physical resources, and iii) the affinity and anti-affinity requirements of VN requests. All these conditions and states of the SN are monitored and updated at each allocation step.

For each allocation step (each received VN request), the objective is to minimize the total occupied bandwidth on the SN. This ensures that the SN utilizes the least amount of bandwidth possible for serving a VN request, which includes the non-obvious requirement of using the shortest available path between two physical nodes.

For all allocation steps (all received VN requests), we use the following metrics to evaluate the global performance of the proposed VNE algorithm:

1) **VN acceptance ratio.** This metric indicates the number of successfully embedded VNs out of the total number of received VN requests, providing a measure for the effectiveness of a VNE approach.
2) **Percentage of used servers and edges.** This metric presents the dynamic occupation of physical nodes and links, monitoring real-time loads on service providers.
3) **Average running time of each successful embed.** This metric records the average time for the proposed VNE to successfully map a VN request, reflecting approach efficiency and performance.

III. Constrained Dynamic VNE Solution

We establish an ILP formulation to propose cost-efficient solutions to map each received VN request. Subsequently, we call this ILP model during each allocation step and update the SN accordingly to achieve globally optimized decisions.

Before investigating the ILP mathematical formulation, we summarize all optimization model parameters and variables.

Parameters of the SN:

- $N^{\mathcal{S}}$: Set of physical nodes in the SN graph.
- $L^{\mathcal{S}}$: Set of physical links in the SN graph.
- $CPU_m^{\mathcal{S}}$: Available CPU of physical node m, where $m \in N^{\mathcal{S}}$.
- $BW_{(m,n)}^{\mathcal{S}}$: Available Bandwidth of physical link (m,n), where $(m,n) \in L^{\mathcal{S}}$ and $m,n \in N^{\mathcal{S}}$.
- $P^{\mathcal{S}}$: Set of shortest available paths in the SN graph.
- $Q_{(m,n)}^{\mathcal{S}}$: Number of physical links included in path (m,n), where $Q_{(m,n)}^{\mathcal{S}} = 1$ denotes a path with one link, while $Q_{(m,n)}^{\mathcal{S}} \geq 2$ refers to a path with more than one link.

Parameters of a VN:

- $N^{\mathcal{V}}$: Set of virtual nodes in a VN graph.
- $L^{\mathcal{V}}$: Set of virtual links in a VN graph.
- $CPU_i^{\mathcal{V}}$: Required CPU of virtual node i, where $i \in N^{\mathcal{V}}$.
- $BW_{(i,j)}^{\mathcal{V}}$: Required Bandwidth by virtual link (i,j), where $(i,j) \in L^{\mathcal{V}}$ and $i,j \in N^{\mathcal{V}}$.
- α_{ij}: Boolean = 0 if nodes i and j of a VN have to be separated; 1 if they should be co-located, where $i,j \in N^{\mathcal{V}}$.
- βn_i: Boolean = 0 if node i does not share physical nodes with other VNs (node anti-affinity); 1 otherwise, where $i \in N^{\mathcal{V}}$.
- $\beta l_{(i,j)}$: Boolean = 0 if link (i,j) does not share physical links with other VNs (link anti-affinity); 1 otherwise, where $(i,j) \in L^{\mathcal{V}}$.

Decision variables:

- x_i^m: Boolean = 1 if virtual node $i \in N^{\mathcal{V}}$ is mapped to physical node $m \in N^{\mathcal{S}}$; 0 otherwise.
- $y_{(i,j)}^{(m,n)}$: Boolean = 1 if virtual link $(i,j) \in L^{\mathcal{V}}$ is mapped to physical path $(m,n) \in P^{\mathcal{S}}$; 0 otherwise.

The objective, in each allocation step, is to provide optimal mappings minimizing the total used bandwidth. This is formulated by (1).

$$\min \sum_{i \in N^{\mathcal{V}}} \sum_{j \in N^{\mathcal{V}}} \sum_{m \in N^{\mathcal{S}}} \sum_{n \in N^{\mathcal{S}}} BW_{(i,j)}^{\mathcal{V}} \cdot Q_{(m,n)}^{\mathcal{S}} \cdot y_{(i,j)}^{(m,n)} \quad (1)$$

The optimization is subject to linear constraints given as:

$$\sum_{m \in N^{\mathcal{S}}} x_i^m = 1, \quad \forall i \in N^{\mathcal{V}} \quad (2)$$

$$\sum_{m \in N^{\mathcal{S}}} \sum_{n \in N^{\mathcal{S}}} y_{(i,j)}^{(m,n)} = 1, \quad \forall (i,j) \in L^{\mathcal{V}} \quad (3)$$

$$\sum_{n \in N^{\mathcal{S}}} y_{(i,j)}^{(m,n)} = x_i^m, \quad \forall (i,j) \in L^{\mathcal{V}}, m \in N^{\mathcal{S}} \quad (4)$$

$$\sum_{m \in N^{\mathcal{S}}} y_{(i,j)}^{(m,n)} = x_j^n, \quad \forall (i,j) \in L^{\mathcal{V}}, n \in N^{\mathcal{S}} \quad (5)$$

$$\sum_{i \in N^{\mathcal{V}}} CPU_i^{\mathcal{V}} \cdot x_i^m \leq CPU_m^{\mathcal{S}}, \quad \forall m \in N^{\mathcal{S}} \quad (6)$$

$$\sum_{i \in N^{\mathcal{V}}} \sum_{j \in N^{\mathcal{V}}} BW_{(i,j)}^{\mathcal{V}} \cdot y_{(i,j)}^{(m,n)} \leq BW_{(m,n)}^{\mathcal{S}}, \forall (m,n) \in P^{\mathcal{S}} \quad (7)$$

$$x_i^m + x_j^m \leq 1 + \alpha_{i,j}, \quad \forall i,j \in N^{\mathcal{V}}, i \neq j, m \in N^{\mathcal{S}} \quad (8)$$

$$x_i^m \geq -1 + x_j^m + \alpha_{i,j}, \quad \forall i,j \in N^{\mathcal{V}}, i \neq j, m \in N^{\mathcal{S}} \quad (9)$$

$$x_i^{\overline{m}} \leq \beta n_i, \quad \forall i \in N^{\mathcal{V}}, \overline{m} \in N^{\mathcal{S}} \tag{10}$$

$$y_{(i,j)}^{(\overline{m},\overline{n})} \leq \beta l_{(i,j)}, \quad \forall (i,j) \in L^{\mathcal{V}}, (\overline{m},\overline{n}) \in P^{\mathcal{S}} \tag{11}$$

$$x_i^m, y_{(i,j)}^{(m,n)} \in \{0,1\} \tag{12}$$

Constraints of (2) require that a virtual node i is hosted by exactly one physical node m. Constraints in (3) ensure that a virtual link (i,j) is served by exactly one physical link (or path) (m,n). A virtual link $(i,j) \in L^{\mathcal{V}}$ is hosted by a physical link/path $(m,n) \in P^{\mathcal{S}}$ such that source virtual node i and destination virtual node j of (i,j) are assigned to physical nodes m and n, respectively. In joint node and link mapping constraints expressed in (4) and (5), if virtual node i is mapped to physical node m, then virtual link (i,j) needs to be mapped to a physical path (m,n) with node m as the source or endpoint, and n as the other endpoint. Constraints (6) and (7) provide guarantees not to exceed the amount of available resource (CPU and Bandwidth in our case) in the physical substrate. Available resources are updated based on the last mapping occupying physical resources and all the VNs departures releasing resources. Constraints (8) and ((9) must be considered together to meet the co-localization requirement of two or more virtual nodes and ensure the separation of different virtual nodes of the same virtual request/graph. Constraints (10) ensure that if a virtual node i has an anti-affinity requirement, then i cannot be hosted by a physical node m that is still hosting other VNs' nodes. Constraints (11) respect link anti-affinity requirements. We use $\overline{m}$ and $(\overline{m},\overline{n})$ in constraints (10) and (11) to denote that a physical node (resp. physical link) is still used to serve other demands. Constraint (12) denotes the range of decision variables.

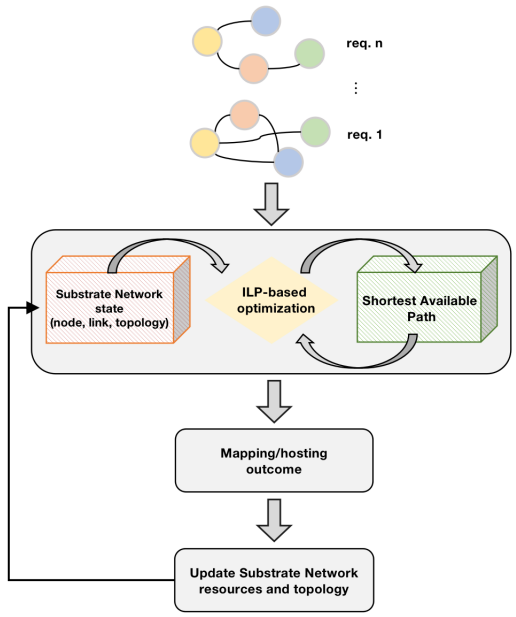

Fig. 2. The ILP-based model for the studied VNE problem.

This ILP model provides the optimal embedding decision of each received VN in each allocation step. For operating all VN requests, we investigate a global ILP-based framework in Fig. 2: after accepting a received VN request, the framework relies on the last update of the SN resources and topology, and solicits the shortest available paths, using a modified Dijkstra algorithm, to optimally provide the embedding decision. Sequentially, it updates the SN information (checking for VN departures releasing resources, last VN placement occupying resources, etc.) before considering the next VN request.

We update the SN resources and topology before each new optimization round using three main steps:

1) Calculate the current available CPU and bandwidth on the SN before the next VN request.
2) Update the SN graph based on the arrivals and departures of VNs. The departures of requests will release free resources, and already hosted VNs occupy a part of the physical resources. Note that releasing isolated physical nodes and links due to anti-affinity constraints changes the SN topology (disconnections in the exploitable part of the SN for mapping occur from time to time).
3) Update the SN graph based on the node/link anti-affinity constraints of the previous virtual request.

IV. Simulation Results

We evaluate the performance of our proposed ILP-based model using randomly generated instances. We solve the exact model using the CPLEX22.1 solver. Our evaluations are conducted on a commercial off-the-shelf PC with an Intel Core CPU of 1.60 GHz and 8 GB RAM.

We use randomly generated instances with VN requests composed of up to 10 nodes and 15 edges. The SN graph is composed of up to 100 nodes and 300 edges. For each node in a given VN request, we randomly generate its required processing capacity in the [5, 20] CPU range. For each physical node in SN, we randomly generate its processing capacity in the [50, 100] CPU range. Similarly, each virtual link has a bandwidth request generated randomly in the [5, 20] Mbps range, while each available physical link has its bandwidth capacity generated in the [50, 100] Mbps range. The simulation runs for up to 10,000 time units depending on simulations, and during this horizon, VN requests may arrive following a Poisson process with an average arrival rate λ of 1 arrival per 50-time units and an exponential service rate μ of 1 departure every 750-time units.

To evaluate the impact of anti-affinity constraints on the VN acceptance ratio, we randomly generate and select a subset of the VN requests to include node/link anti-affinity requirements. We use the generic form **[rate of node anti-affinity, rate of link anti-affinity]** to indicate the level of node anti-affinity and link anti-affinity, respectively. We consider 4 levels of affinity and anti-affinity scenarios as follows: i) [high, high], ii) [high, low], iii) [low, high], and iv) [low, low]. This notation can be explained with, for instance, the [high, low] case that corresponds to VN request arrivals with a high proportion of node anti-affinity constraints (80% in

the simulation) and a low proportion of link anti-affinity constraints (20% in the simulation). Fig. 3 shows the obtained average VN acceptance ratios in different simulations.

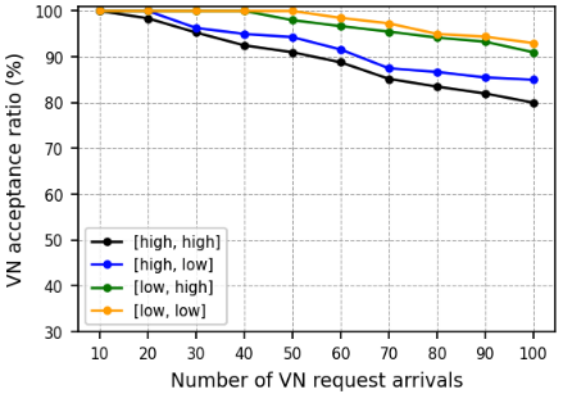

Fig. 3. Impact of node/link anti-affinity on VN acceptance ratios.

Fig. 3 depicts the impact of node and link anti-affinity constraints on VN acceptance ratios. For VN requests with a high proportion of anti-affinity constraints and requirements (i.e., the [high, high] scenario), the achieved acceptance ratio is only around 80% for 100 received VN requests. For much less constrained VN requests corresponding to the [low, low] scenario, the acceptance rate is as expected much higher and around 93% for 100 VN arrivals. This shows the efficiency of our proposed algorithm that manages in constrained conditions, where nodes and links are not available for hosting due to separation and isolation constraints in and across VNs, to find hosting solutions and to continue consolidating the SN resources and topology. The scenario [high, low] considering a **high** number of nodes anti-affinity constraints, and a **low** number of links anti-affinity constraints, confirms how difficult it is to achieve a high acceptance ratio (close to 85%, in the worst case) due to the disconnections of the SN graph or topology leading to reject more requests. In the scenario [low, high] (with a high number of link anti-affinity constraints), however, the algorithm rejects fewer VN requests thanks to the shortest path algorithm that manages to find other solutions in the SN available graph resources. The results also indicate that node anti-affinity constraints, in the scenarios [high, X] (where X can be "high", or "low"), have the highest impact on decreasing VN acceptance ratios because of the isolation requirements, some hosting nodes become unusable and, depending on their node degrees, make associated links unusable as well.

To further assess the impact of anti-affinity constraints on service providers, we collect in another simulation the percentage of additional servers and links that are needed to host virtual requests when compared with a scenario without any isolation requirements. The results reported in Fig. 4 correspond to a simulation scenario with [high, high] anti-affinity conditions and a SN composed of 50 nodes and 94 links confronted to dynamically arriving and departing VN requests. The maximum values of the ordinates in the two sub graphs correspond to the hosting percentage of physical servers and edges, respectively. The simulation duration is 5000 time units. The results depicted in Fig. 4 have to be analyzed consequently by focusing on the maximum load interval in the simulation, namely in the range $[2500, 3500]$ time units. In order to accommodate VN requests with harder anti-affinity constraints and requirements, a cloud service provider will have to invest or use more physical servers and links. Without any loss of generality, even if the simulated scenario is arbitrary, we observe that 85% physical nodes have to be used instead of 60% nodes without any constraints. Similarly, more links, actually up to 57% links, need to be used instead of 46% links when comparing VN requests with anti-affinity constraints with requests that do not have anti-affinity requirements. These results highlight the need for a provider to invest or operate more servers and links in order to serve users with stringent separation and isolation requirements. The algorithm proposed in this work enable providers to estimate, dimension and plan their infrastructures accordingly by running the algorithm for their specific scenarios based on (precise or estimated) knowledge of their clients typical requirements and demands.

Fig. 5 provides insight into how the proposed algorithm scales with the problem size in terms of the increasing number of VN requests and for SN graph sizes of 50 and 100 nodes. Since we are relying on an underlying ILP formulation, complexity is high and resolution time increases exponentially with increasing size. Speeding resolution time has not been an objective in this paper since the focus is on really understanding and assessing the impact of anti-affinity requirements expressed by users on the provider so the latter can anticipate the number of additional physical nodes and links they must acquire, make available and operate compared with a situation where clients do not have any constraints (corresponding to a lower bound in the SN size actually).

V. Conclusions and Perspectives

We address a constrained dynamic virtual network embedding problem that integrates clients affinity and anti-affinity requirements on nodes and links. These requirements apply not only within VN requests, but also across VN requests from different clients. We formulate an integer-linear programming model to place virtual nodes and links jointly and get the optimal mapping for each virtual network onto the dynamically updated physical network. We report performance to provide insight to cloud service providers on additional investments in nodes and links they should make to serve clients with affinity and anti-affinity requirements. They can estimate this need by running our proposed VNE algorithm on their specific scenarios. Future works to enhance this algorithm, and make it even more useful to providers that serve users with heterogeneous demands, include: (i) treating the requests in batch and queuing rejected VN requests for additional attempts to reduce the rejection rate and (ii) taking into account uncertainty by adopting a stochastic optimization approach by adding chance constraints to our ILP-based algorithm.

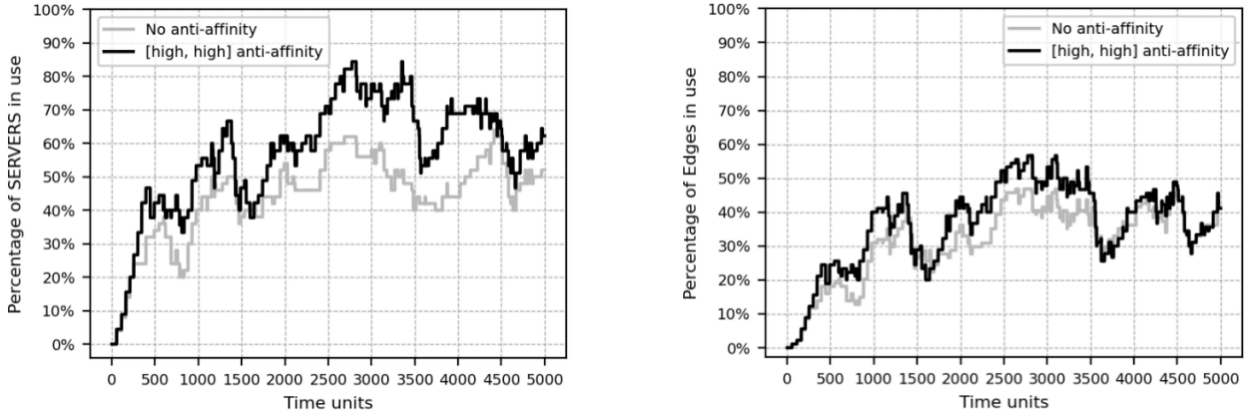

Fig. 4. Percentage of servers and links used by VN requests with [high, high] anti-affinity and without anti-affinity requirements.

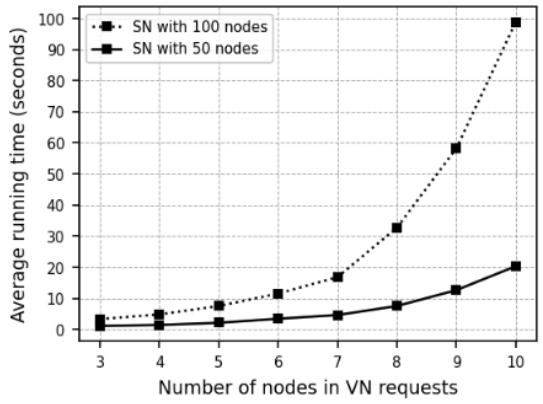

Fig. 5. Convergence time under different topology sizes.

REFERENCES

[1] H., Cao, H., Zhu, and L., Yang, "Collaborative attributes and resources for single-stage virtual network mapping in network virtualization," Journal of Communications and Networks, 2019, vol. 22, pp. 61–71.

[2] A., Fischer, J. F., Botero, M. T., Beck, H., De Meer, and X., Hesselbach, "Virtual network embedding: A survey," IEEE Communications Surveys & Tutorials, 2013, vol. 15, pp. 1888–1906.

[3] Z., Kotulski, T. W., Nowak, M., Sepczuk, and M. A., Tunia, "5G networks: Types of isolation and their parameters in RAN and CN slices," Computer Networks, 2020, vol. 171, pp. 107135.

[4] M., Mechtri, M., Hadji, and D., Zeghlache, "Exact and heuristic resource mapping algorithms for distributed and hybrid clouds," IEEE Transactions on Cloud Computing, 2017, vol. 5, pp. 681–696.

[5] Z., Li, Z., Lu, S., Deng, and X., Gao, "A self-adaptive virtual network embedding algorithm based on software-defined networks," IEEE Transactions on Network and Service Management, 2018, vol. 16, pp. 362–373.

[6] Z., Allybokus, N., Perrot, J., Leguay, L., Maggi, and E., Gourdin, "Virtual function placement for service chaining with partial orders and antiaffinity rules," Networks, vol. 71, pp. 97–106.

[7] M., Gao, B., Addis, M., Bouet, and S., Secci, "Optimal orchestration of virtual network functions," Computer Networks, 2018, vol. 142, pp. 108–127.

[8] A. S., Jacobs, R. J., Pfitscher, R. L., dos Santos, M. F., Franco, E. J., Scheid, and L. Z., Granville, "Artificial neural network model to predict affinity for virtual network functions," in proceeding of NOMS 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, April 23-27, 2018, pp. 1–9.

[9] X., Shen, Q., Dai, S., Mao, F., Chung, and K. S., Choi, "Network together: Node classification via cross-network deep network embedding," IEEE Transactions on Neural Networks and Learning Systems, 2020, vol. 32, pp. 1935–1948.

[10] C., Mei, J., Liu, J., Li, L., Zhang, and M., Shao, "5G network slices embedding with sharable virtual network functions," Journal of Communications and Networks, 2020, vol. 22, pp. 415–427.

[11] N., Bouten, M., Claeys, R., Mijumbi, J., Famaey, S., Latré, and J., Serrat, "Semantic validation of affinity constrained service function chain requests," in proceedings of IEEE NetSoft conference and workshops (NetSoft), Seoul, Korea, June 06-10, 2016, pp. 202–210.

[12] N., Bouten, R., Mijumbi, J., Serrat, J., Famaey, S., Latré, and F., De Turck, "Semantically enhanced mapping algorithm for affinity-constrained service function chain requests," IEEE Transactions on Network and Service Management, 2017, vol. 14, pp. 317–331.

[13] A., Fischer, D., Bhamare, and A., Kassler, "On the construction of optimal embedding problems for delay-sensitive service function chains," in proceedings of the 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, July 29 - August 01, 2019, pp. 1–10.

[14] F., He and E., Oki, "Shared protection-based virtual network embedding over elastic optical networks," IEEE Transactions on Network and Service Management, 2022, vol. 19, pp. 2869–2884.

[15] T., He, K. W., Chin, H., Ren, and X., Liu, "Maximizing virtual network embedding requests in RF-charging IoT networks." IEEE Communications Letters, 2022 vol. 26, pp. 863-–867.

[16] H., Cao, Y., Zhu, G., Zheng, and L., Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," IEEE Transactions on Network and Service Management, 2017, vol. 15, pp. 356–371.

[17] C., Aguilar and J., Rubio, "A novel evaluation function for higher acceptance rates and more profitable metaheuristic-based online virtual network embedding," Computer Networks, 2021, vol. 195, pp. 108191.

[18] 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 Transactions on Networking, 2016, vol. 24, pp. 3648–3661.

[19] S., Wang, J., Bi, J., Wu, A. V., Vasilakos, and Q., Fan, "VNE-TD: A virtual network embedding algorithm based on temporal-difference learning," Computer Networks, 2019, vol. 161, pp. 251–263.

[20] P., Zhang, C., Wang, N., Kumar, W., Zhang, and L., Liu, "Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning," IEEE Internet of Things Journal, 2021, vol. 9, pp. 9389–9398.

[21] W., Zhang, D., Wang, S., Yu, H., He, and Y., Wang, "Repeatable multi-dimensional virtual network embedding in cloud service platform," IEEE Transactions on Services Computing, 2021, vol. 15, pp. 3499–3512.

[22] S., Khebbache, M., Hadji, and D., Zeghlache, "Virtualized network functions chaining and routing algorithms." Computer Networks, 2017, vol. 114, pp. 95–110.