A Parallel Link Mapping for Virtual Network Embedding with Joint Load-Balancing and Energy-Saving

Mario Minardi, Shree Krishna Sharma, Symeon Chatzinotas and Thang X. Vu
Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
email address: {mario.minardi, shree.sharma, symeon.chatzinotas, thang.vu}@uni.lu

***Abstract*—Virtual Network Embedding (VNE) is an emerging area of Telecom Networks, in an era where the physical capacity of the substrate network is pushed to the limits in order to get the maximum achievable performance. Link mapping optimization is one of the two sub-problems which the VNE is subdivided into. The objective of the paper is to demonstrate the efficiency of a parallel approach with respect to the sequential one, in particular when there is a high level of heterogeneity among the Virtual Networks to be embedded, such as network slicing scenario. In this regard, this paper investigates the significance of a parallel approach in effectively implementing VNE. Specifically, we focus the attention to the link mapping, considering the node mapping is already known a priori. In contrast to the trend in the literature to use Genetic Algorithms (GAs) for a parallel computation, this paper proposes a novel and efficient parallel link mapping algorithm for enabling VNE in realistic system scenarios. Two important objectives are considered, load-balancing and energy-saving, and the presented results demonstrate the superiority of the proposed parallel approach over the sequential one in terms of these objectives, at the expenses of the computing time. Furthermore, the scalability of the proposed algorithm is evaluated over a range of substrate network sizes.**

I. Introduction

Nowadays, the complexity and heterogeneity of the upcoming telecommunication networks are pushing towards the research of new optimization techniques/strategies to effectively maximize the utilization of the physical network, also defined as substrate network. The large variety of applications and services has pushed network providers to look for efficient network virtualization techniques, with the aim of supporting heterogeneous networks/services within the same substrate network.

In the above context, network slicing has gained a lot of success since it aims to allocate several end-to-end services ("slices") in the same substrate network [1]. Basically, very different services are thought to run onto the same substrate network and each one of them requires its own pool of resources that have to be allocated. The challenging allocation of a service/network in the physical network is defined as Virtual Network Embedding (VNE) [2]. VNE is the mapping of an incoming Virtual Network Request (VNR), or multitude of them, onto the substrate resources. A VNR is a network composed by nodes, links and, eventually, requirements for both links and nodes. Therefore, VNE problem is based on two sub-mappings, node and link mapping.

VNE is an NP-Hard problem, hence, a trade-off between complexity and optimality always comes into play. In the literature, a large variety of solutions have been presented, trying to reduce the complexity of the problem, and therefore the computing time, while providing a solution as closer as possible to the optimal one [3]–[6]. This paper studies the efficiency of a parallel Link Mapping computation for the VNE problem. In order to avoid any confusion in the terminology, it is worth underlining that, in this paper, the term *sequential* and *parallel* refer to how a set of virtual network requests is embedded, rather than the order of node mapping and link mapping computation, as it usually presented in the literature. Therefore, *sequential* approach, here, means that one VNR is embedded per time. On the contrary, in *parallel* approach, a set of VNRs are embedded each iteration.

The following two subsections introduce the related works and the main contributions of this paper.

A. Related Works

Among significant research works on VNE in the literature, three main trends are summarized below and used as motivation for the contributions of this paper.

The optimization of the node mapping procedure is one of the most common approaches. Once the assigned substrate node(s) are defined, the virtual links are embedded, most of the time, with k-shortest paths or multi-commodity flow algorithm [3], [4], [7]–[9]. In order to get closer to the optimal solution, the augmentation technique, proposed in [5], has been widely utilized in the literature [3], [10]. Considering a set of VNRs to be embedded together by this approach, the complexity of the problem may be practically infeasible. In addition, shortest-path algorithms do not always provide an optimal solution in term of metrics different from the path length.

The second motivation of this work comes from the fact that a sequential approach impacts the performance of the solution because the solution is strongly dependent on the order which the VNRs are elaborated into. The sequential approach is usually derived from the nature of the problem itself, since in practice, VNRs enter and leave the network at different instants of time. In the network slicing era, the heterogeneity of services' requirements has brought about the need to look for QoS-oriented parallel solutions. Despite the increase of the complexity, the parallel VNE problem formulation has

become extremely relevant for the development of an efficient services' mapping. This contribution wants to show how the performance in an heterogeneous scenario can be improved when there is a parallel embedding of VNRs.

At this point, the third motivation arises because in the literature, there are some works dealing with both the link mapping optimization and the performance improvement due to a parallel approach. Authors of [6], [11] focused on the relevance of the link mapping optimization rather than the node mapping, and proposed a parallel approach based on the Genetic Algorithm (GA). This parallel method, however, only considers the link mapping and is able to embed one VNR at a time. The proposal presented in [12] goes one step further, where authors proposed the usage of GA for a parallel link mapping applied to a set of VNRs instead of just one. The idea is that the parallel computation can increase the coordination among all VNRs embeddings and, consequently, achieve better performance. In the literature, among the parallel approaches proposed, GA seems to be the most promising one because it allows to reduce the computing time, through parallel computation, and, at the same time, to improve the solution. The computing time is reduced due to the fact that parallel machines, exploiting a priori knowledge, are initially fed with a set of link mappings (typically k-shortest paths), from which they produce new feasible solutions.

However, a priori knowledge might not always be available. If we consider for example dynamic networks (e.g. low-earth orbit satellite networks), online embedding algorithms might be required in real-time, especially if the connectivity variations are not predictable, due to the unavailability of some embeddings.

B. Contributions

With regard to the three research trends identified in Sec. I-A, our contribution is three-fold:

- We propose an Integer Linear Program (ILP) formulation for the link mapping for the VNE problem, while considering the utility function combining Load-Balancing and Energy-Saving objectives.
- We illustrate how the increase in the level of parallelism in the system corresponds to a higher quality embedding. The average substrate link utilization and the energy consumed by active substrate nodes are the two metrics considered to evaluate the efficiency of the algorithm.
- We also show how the parallelism approach can perform better than the sequential one when there is higher level of heterogeneity in the network.

The remainder of the paper is organized as follows. Section II presents the Network Model and the formulation of the problem. In Section III, the steps of the proposed VNE algorithm are illustrated and Section IV presents the performance evaluation. Finally, Section V concludes the paper.

II. Network Model and Problem Description

Fig. 1 shows two VNRs, with their respective resource requirements, which are mapped onto the same substrate network. In the traditional approach, the algorithm is computed every time a new VNR is requested for embedding.
Usually, the problem is initiated with both the actual status of the substrate network and the VNR, and the embedding for that specific network is provided. Then, the actual substrate network will be updated and the algorithm will run again for the following VNR. It is widely known [6], [11], [12] that this approach can achieve only a local near-optimal solution since the optimization is applied independently for each VNR.

On the contrary, considering the simultaneous embedding of a set of VNRs, would bring to a global near-optimal solution. This concept is at the basis of this contribution, where we would like to demonstrate the advantages of a parallel approach.

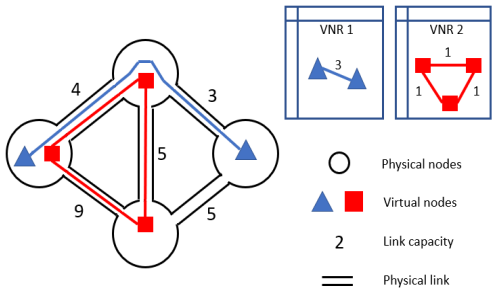

Figure 1. Illustration of Virtual Network Embedding

A. Problem Initialization

The substrate network is modelled as a directed weighted graph: $G_s = (N_s, E_s)$, with N_s the set of substrate nodes and E_s the set of substrate edges. Every substrate node n_s is assigned with the power consumption $p(n_s)$ and every substrate link (u, v) is assigned with the capacity $c(u, v)$. p_{sum} represents the sum of all the substrate nodes' power consumption. The general n-*th* VNR is modelled as a directed graph: $G_v^n = (N_v^n, E_v^n)$, with N_v^n the set of virtual nodes and E_v^n the set of virtual edges.

For every virtual link $i \in E_v^n$, the demanded bandwidth $bw(i, n)$ is defined. In addition, the node mapping is considered already known as a priori, hence, s_i^n and d_i^n are the source and destination substrate nodes of the i-*th* link of the n-*th* VNR, respectively. The node mapping is initially computed via the D-ViNE approach proposed in [5].

It is important to clarify that this step has been done in order to get feasible node mappings. It could have been completely random, but unfeasible solutions could have come into play and the purpose of this work is to consider scenarios where there is always a feasible solution. Finally, the set *VN* represents all the VNRs to be embedded.

B. Problem Formulation

In the following, the objective and constraints of the problem are presented. The objective function (1) represents the combination of load-balancing and energy-saving objectives. The load-balancing objective aims to spread the traffic in the

substrate network as efficiently as possible. To do so, it needs to minimize the overall load in each substrate link. On the contrary, the energy-saving minimizes the amount of power consumed in the network by all the active substrate nodes. The combined objective can be written as:

$$\min_{z_{uv}^{i,n}, y_{n_s}} (\alpha \cdot \sum_{n \in VN} \sum_{uv \in E_s} \sum_{i} \frac{z_{uv}^{i,n} \cdot bw(i,n)}{c(u,v) + \epsilon} + \beta \cdot \frac{1}{p_{sum}} \cdot (\sum_{n_s \in N_s} y_{n_s} \cdot p(n_s))), \quad (1)$$

where the first term represents the load-balancing objective and the second term defines the energy-saving objective. The parameters α and β, are the weights of the two objectives. A substrate node, hosting one or multiple virtual nodes, is considered active when traffic is passing through it. Therefore, with the energy-saving objective, more priority is given to the less power-consuming nodes in order to reduce the sum of the total consumption.

Table 1: Variables in eq. (1) - (10)

$f_{uv}^{i,n}$	flow of the i-*th* virtual link of n-*th* VNR on the substrate link $uv \in E_s$
y_{n_s}	binary variable to indicate an active node
$z_{uv}^{i,n}$	binary variable to indicate if the i-*th* link of n-*th* VNR is embedded in uv

While load-balancing objective spreads the traffic, energy-saving tends to concentrate it in order to reduce the amount of active nodes. In the following, the constraints of the problem are formulated and explained.

$$\sum_{n \in VN} \sum_{i \in E_v^n} f_{uv}^{i,n} \leq c(u,v), \quad \forall (u,v) \in E_s \quad (2)$$

$$\sum_{w \in N_s} f_{s_i^n w}^{i,n} - \sum_{w \in N_s} f_{w s_i^n}^{i,n} = bw(i,n), \quad \forall i, \forall n \quad (3)$$

$$\sum_{w \in N_s} f_{w t_i^n}^{i,n} - \sum_{w \in N_s} f_{t_i^n w}^{i,n} = -bw(i,n), \quad \forall i, \forall n \quad (4)$$

$$\sum_{v \in N_s} f_{vu}^{i,n} - \sum_{v \in N_s} f_{uv}^{i,n} = 0, \quad \forall i, \forall n, \forall u \in N_s \setminus \{s_i^n, d_i^n\} \quad (5)$$

$$f_{uv}^{i,n} \geq 0, \quad \forall (u,v) \in E_s, \forall i, \forall n \quad (6)$$

$$z_{uv}^{i,n} = \frac{f_{uv}^{i,n}}{bw(i,n)}, \quad \forall i, \forall n, \forall (u,v) \in E_s \quad (7)$$

$$z_{uv}^{i,n} \in \{0,1\}, \quad \forall (u,v) \in E_s, \forall i, \forall n, \quad (8)$$

$$y_{n_s} = \begin{cases} 1, & \text{if } \sum_i \sum_n f_{u n_s}^{i,n} > 0 \text{ or } \sum_i \sum_n f_{n_s u}^{i,n} > 0 \\ 0, & otherwise \end{cases} \quad (9)$$

$$z_{uv}^{i,n} + z_{vu}^{i,n} \leq 1, \quad \forall (u,v) \in E_s, \forall i, \forall n, \quad (10)$$

In (2), for each substrate link (u,v), the sum of the bandwidth occupied by all virtual links mapped in (u,v), is upper-bounded by the maximum capacity of the substrate link.
Constraints (3) and (4) ensure that for each virtual link i and for every VNR, given its source and destination substrate nodes, the total flow outgoing from the source s_i^n is equal to the bandwidth required and the total flow incoming in the destination t_i^n is equal to the required bandwidth with a negative sign.
(5) states that given the i-*th* virtual link of the n-*th* VNR, for every intermediate node between its source and destination, the sum of incoming flows has to be equal to the sum of outgoing flows (flow's conservation law).
In (6), the flow variable is set to be positive. Equations (7) and (8) show the flow and link mapping variables definition. (9) defines the binary variable which determines if a substrate node is active or not.
Finally, (10) avoids unwanted loops. Indeed, for each substrate link and for each virtual link i of the n-*th* VNR, the flow can go either in one direction or in the opposite one.

III. Proposed Parallel Link Mapping Algorithm

In this section, the steps of the proposed algorithm are described.

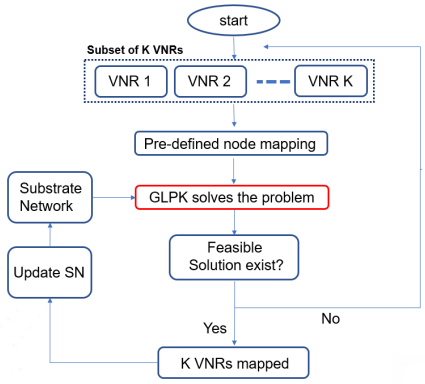

Figure 2. Illustration of the proposed parallel link mapping algorithm

The set *VN* is composed by N VNRs. As highlighted among the motivations, the aim of this paper is to propose and illustrate the gain of a parallel approach with respect to the sequential one. The flow chart in Fig. 2 shows the parallel approach. From the main set *VN*, smaller subsets of K VNRs are created and given as input to the algorithm (K represents the parallelism level). The problem is compiled and executed from GNU Linear Programming Kit (GLPK) solver, which looks for the solution of a K-VNRs embedding simultaneously. The algorithm starts with the creation of the substrate graph, in Matlab. Then, Matlab generates the VNRs, with the bandwidth and the position of the two end-nodes in the network (pre-defined node mapping). A file is initiated with all this data and the GLPK solver is called.

Table 2: Parallel Link Mapping Algorithm

1: **Begin**
2: Generation of substrate graph according to (11)
3: Generation of the set of VNRs randomly (Table 3)
4: pre-computed node mapping with D-ViNE algorithm
5: subdivision in subsets with K VNRs
6:
7: **Input**
8: Objective function
9: substrate link capacity $c(u,v), \forall (u,v) \in E_s$
10: Power consumption $p(n_s), \quad \forall n_s \in N_s$
11: $s_i^n, d_i^n, bw(i,n), \quad \forall n \in VN, \forall i \in E_v^n,$
12: **Output**
13: Link mapping per each VNR
14:
15: **If** branch-and-cut method finds a solution
16: Matlab updates the substrate resources
17: **else** Link mapping failed
18: **End**

In the solver, the problem is formulated following the constraints and objectives (1)-(10). GLPK looks for the solution utilizing the branch-and-cut method. If there is a solution, GLPK will return the results back to Matlab to update the substrate network resources and compute the performance required. In total, the algorithm runs for N/K times. The pseudo-code is presented in Table 2.

IV. Performance Evaluation

A. Simulation setup

The nodes of the substrate network are randomly distributed in a 100 x 100 grid. Given u and v pair of substrate nodes, the probability to be connected is given by:

$$P(u,v) = \gamma \cdot \exp(-\frac{d}{\delta \cdot d_{max}}) \tag{11}$$

where d is the geometric distance in the grid between the two nodes and d_{max} is the maximum distance between any pair of substrate nodes. In addition, γ and δ are two design parameters which allow to control the complexity of the network. We use $\gamma = \delta = 0.5$, as the "*simple case*" considered in [3]. The capacity of each substrate link is randomly generated in the interval [60, 80]. Each VNR is composed by 2 virtual nodes and 1 virtual link. The virtual bandwidth required from the generic n-*th* VNR is a random number in a defined interval. Different cases of this interval are considered (Table 3). Finally, the cardinality N of *VN* is set to 30.

B. Evaluation Results

Matlab and GLPK solver are used for the simulations. The substrate network for the results in Fig. 1-3 is composed by 30 nodes. The scenario in Fig. 4 considers higher number of substrate nodes to demonstrate the scalability of the algorithm. In the first two case studies, we analyze the performance of the algorithm with the variation of the parallelism level K. The two extremes are the sequential case, with K = 1, and the fully-parallel one with K = 30.

1) Parallelism vs Heterogeneity

We study the efficiency of the parallelism approach, looking at different levels of heterogeneity for the required bandwidth of VNRs. Three different cases are considered (Table 3).

Table 3: Virtual Networks demanded bandwidth

	[min, max]
Case 1	[5, 20]
Case 2	[5, 30]
Case 3	[5, 40]

Fig. 3 shows how the parallelism efficiency increases with the increase in the range of virtual bandwidth demanded. This is due to the fact that if there is more heterogeneity in the demanded bandwidth, the algorithm is able to better manage the link mapping in order to achieve the objective with respect to the sequential approach. Therefore, the VNR which demands for more bandwidth will be embedded in the path with more available bandwidth. For example, for Case 1, since there is almost no difference among all VNRs, the efficiency of fully parallel approach compared to the sequential case is almost none. While in Case 3, the fully parallel case performs much better. Load-Balancing objective is considered for this scenario, with average substrate link utilization as the metric.

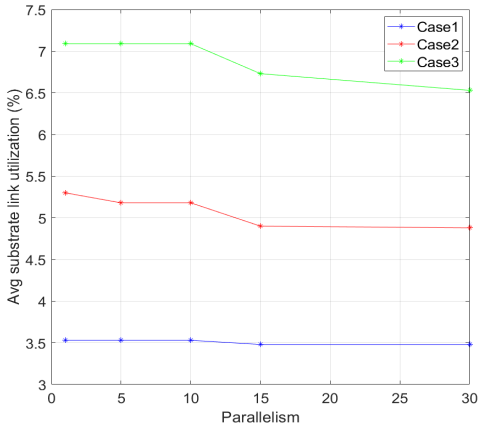

Figure 3. Average substrate utilization versus parallelism with different heterogeneity levels

2) Parallelism vs Computing Time

The complexity of the problem increases with the increase in the level of parallelism (K in Fig. 2). Consequently, as shown in Fig. 4, the more the VNRs are embedded per time, the higher computing time will be required. We also apply the three cases in Table 3 for this scenario, and present the corresponding results in Fig. 4. It can be noticed that when the parallelism level is high, the complexity of the problem is very sensitive to the load of the network. Indeed, the difference in Case 3 (highest load) from those of Cases 1 and 2 is quite significant when the parallelism level is the maximum, i.e., 30. For each case, the main trend is that the computing time, and consequently the complexity, increases exponentially with the increase in the level of the parallelism.

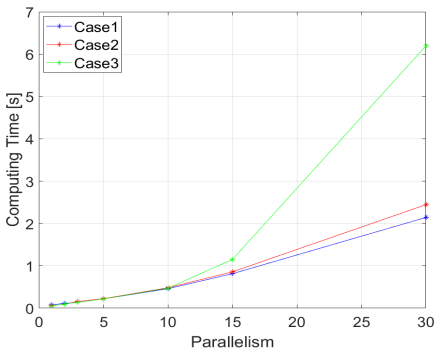

Figure 4. Computing time vs parallelism

3) Load-Balancing and Energy-Saving combination

This subsection aims to compare the results with different weights given to the objective function (1). The value α, and consequently β, varies from 0 to 1, with intervals of 0.25. α = 1 (β = 0) represents the Load-Balancing objective. On the other hand, α = 0 (β = 1) depicts the Energy-Saving objective.

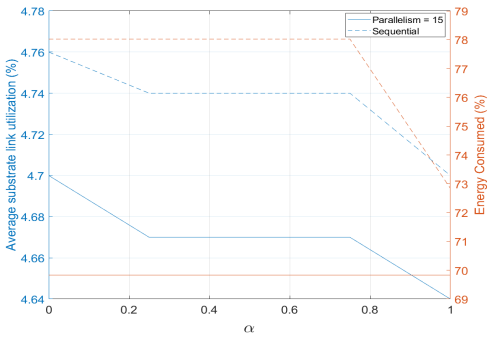

Figure 5. Average substrate link utilization and Energy consumed versus α

Intermediate values represent a weighted combination of those two objectives. A level of parallelism is fixed to 15 and compared to the sequential case, with average substrate link utilization and energy consumption as metrics.

For both metrics, the parallel approach (solid lines) always performs better (lower values) than the sequential case (dashed lines). In addition, the negligible effect of α on energy consumption for the considered case of parallelism = 15 is due to the dominance of the second term over the first term in (1).

4) Scalability

This subsection focuses on the scalability of the algorithm. The algorithm is tested with higher number of substrate nodes (Fig. 6). We fix the level of parallelism to 15 and we compute the algorithm with $\alpha = 1$ and $\beta = 0$ (Load-Balancing case). The average link utilization (left y-axis) decreases with the increase in the number of nodes because the traffic to be embedded is still the same but the available substrate resources are more due to larger size of the physical network.

On the opposite, the computing time (right y-axis) increases due to the increased size of the network, which raises the complexity of the problem.

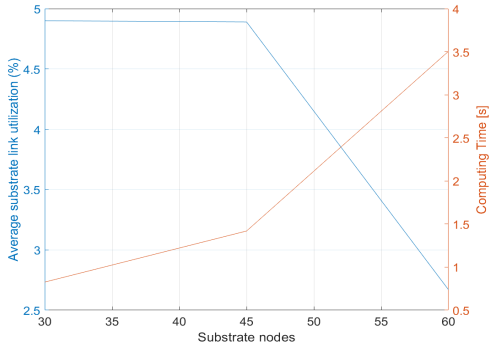

Figure 6. Average substrate link utilization and computing time versus the number of substrate nodes

V. Conclusion

In this paper, we proposed an ILP formulation for the link mapping problem involved in implementing VNE in dynamic networks. The main objective is to highlight the advantages of a parallel approach, over a sequential one, with Load-Balancing and Energy-Saving as the performance objectives of the network. The presented results illustrated how a parallel approach always brings to a more efficient solution with respect to the sequential one, especially when there is a high level of heterogeneity among virtual networks. However, the computing time analysis for different level of parallelism depicts that the fully parallel approach might not always be computed in a limited amount of time if the number of VNRs or the size of substrate network is high. Limited-size but dynamic networks, such as satellite networks, might be the suitable scenario for this approach. In addition, it has been shown that already intermediate values of parallelism level can enhance both the considered performance objectives, while keeping the computing time reasonably lower. Finally, the scalability of the proposed VNE algorithm was illustrated over a range of substrate network sizes.

VI. Acknowledgment

This work is supported by Fond National de la Recherche Luxembourg (FNR) programme, Industrial Partnership Block Grant (IPBG) and the FNR project, Autonomous Network Slicing for Integrated Satellite-Terrestrial Transport Networks (ASWELL). We would like to thank Dr. Christos Politis from SES for following the progress.

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429 – 2453, 3rd Quart. 2018.

[2] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888 – 1906, 4th Quart. 2013.

[3] M. Pham, D. B. Hoang, and Z. Chaczko, "Congestion-aware and energy-aware virtual network embedding," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 210 – 223, Feb 2020.

[4] 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*, vol. 15, no. 1, pp. 356 – 371, Nov. 2017.

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

[6] K. T. Nguyen and C. Huang, "An intelligent parallel algorithm for online virtual network embedding," *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, August 2019.

[7] Y. Xue, J. Peng, K. Han, and Z. Zhu, "On table resource virtualization and network slicing in programmable data plane," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 319 – 331, March 2020.

[8] Z. Yan, N. Wei, Q. Jin, and X. Zhou, "Latency-aware resource-efficient virtual network embedding in sdn," *The 28th Wireless and Optical Communication Conference (WOCC 2019)*, May 2019.

[9] J. Liu, X. He, T. Chen, X. Wang, R. Luo, and T. Huang, "Sn-vne: A virtual network embedding algorithm for satellite networks," *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, August 2019.

[10] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," *IEEE INFOCOM 2009*, April 2009.

[11] Q. Lu and C. Huang, "Distributed parallel vn embedding based on genetic algorithm," *2019 IEEE Symposium on Computers and Communications (ISCC)*, July 2019.

[12] Z. Zhou, X. Chang, Y. Yang, and L. Li, "Resource-aware virtual network parallel embedding based on genetic algorithm," *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Dec 2016.