Using integer programming to embed large virtual networks

Amal Benhamiche[1]   Pierre Fouilhoux[2]   Lucas Létocart[2]   Nancy Perrot[2]   Alexis Schneider[12]

***Abstract*— Virtual Network Embedding (VNE) is an optimization problem at the core of many modern network telecommunication technologies related to the implementation of virtual networks, such as Network Slicing. The VNE problem consists in finding an optimal assignment of virtual demands to physical resources, encompassing simultaneous placement and routing decisions.**

We study the offline version of the VNE, which arises in the context of decision-making for resource allocation and network slice planning. For large networks, the heuristics of the literature often struggle to find solutions, especially when available resources (on nodes and edges) are sparse.

To address these challenges, we explore mathematical programming approaches. Since the classical Flow Formulation provides a weak linear relaxation, we consider a novel formulation, based on a partition of the virtual graph into smaller virtual subgraphs. Since this formulation has an exponential number of variable, its linear relaxation can be solved with Column Generation. We devise a Price-Branch heuristic able to solve large instances, while providing optimality gap. The resulting computational experiments indicate our Price-Branch heuristic is often the only algorithm able to find a solution from a certain instance size, largely outperforming the Flow Formulation or literature heuristics.

I. Introduction

Telecommunication operators (telcos) must prepare for the deployment of an increasing number of virtual networks to meet customer demands for tailored services. Several promising networking technologies leverage common physical resources to host multiple virtual components spanning across the whole network. Notably, Network Slicing enables the creation of virtual networks specifically designed for the requirements of diverse services, such as low latency for Ultra-Reliable Low-Latency Communication (URLLC) applications or high bandwidth for Enhanced Mobile Broadband (eMBB). This concept is central to the vision of 5G networks and allows telcos and Infrastructure Providers to manage resources with enhanced flexibility and programmability while also improving quality of service [1], [2].

The long-term allocation of network slices introduces a new dimension to resource management, transforming it from a purely reactive task, as for traditional packet routing, into a complex planning challenge. Moreover, in the considered realistic scenarios, these network slices can be large-scale, spanning multiple cities and countries, requiring the development of advanced planning strategies and resource management mechanisms.

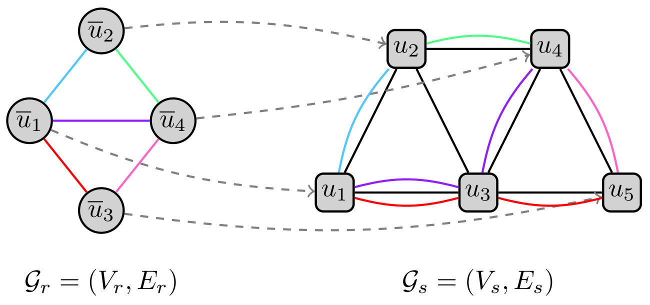

Fig. 1. Example of an embedding of a virtual network (on the left) on a substrate network (on the right)

In this context, achieving an allocation of physical network resources that effectively meets heterogeneous slice requests while minimizing deployment costs represents a key aspect for telcos and the planning of their future networks. The underlying optimization challenge behind is known as the Virtual Network Embedding (VNE) problem, a combinatorial graph problem that requires the simultaneous allocation of physical computing and bandwidth capabilities for the services.

A. Problem Statement

The physical (or substrate) network is represented by an undirected graph $\mathcal{G}_s = (V_s, E_s)$ offering resources on which the (undirected) virtual network $\mathcal{G}_r = (V_r, E_r)$ is to be embedded. The allocation of resources consists in finding a mapping between physical and virtual networks that is defined as follows.

Definition 1 (Mapping). *A mapping $m = (m_V, m_E)$ of $\mathcal{G}_r$ on $\mathcal{G}_s$ is a pair of functions, with:*
$m_V : V_r \to V_s$, the node placement, and
$m_E : E_r \to P_s$, the edge routing, where P_s is the set of simple paths of $\mathcal{G}_s$, and for $\overline{e} = (\overline{u}, \overline{v}) \in E_r$, $m_E(\overline{e})$ is a path of $\mathcal{G}_s$ connecting $m_V(\overline{u})$ and $m_V(\overline{v})$.

Figure 1 illustrates the embedding of a virtual graph on the left over a substrate graph on the right. The dotted arrows show the placement of virtual nodes. A virtual edge of a given color is routed using a substrate path of the same color. In this example, $m_V(\overline{u}_1) = u_1$, $m_V(\overline{u}_3) = u_5$ and $m_E(\overline{u}_1, \overline{u}_3) = \{(u_1, u_3), (u_3, u_5)\}$.

Substrate nodes (resp. edges) have a computational capacity $c_u \in \mathbb{N}_+$ for $u \in V_s$ (resp. bandwidth $c_e \in \mathbb{N}_+$ for $e \in E_s$). Similarly, the virtual nodes and edges have demands. However, since they are not well-known yet, we

[1]Amal Benhamiche, Nancy Perrot and Alexis Schneider are with Orange Innovation, F-92320 Châtillon, France (e-mails: amal.benhamiche, nancy.perrot, alexis.schneider@orange.com).

[2] Pierre Fouilhoux, Lucas Létocart and Alexis Schneider are with Université Sorbonne Paris Nord, CNRS, Laboratoire d'Informatique de Paris Nord, LIPN, F-93430 Villetaneuse, France (e-mails: pierre.fouilhoux, lucas.letocart@lipn.univ-paris13.fr).

will for simplicity consider uniform demands (which is equivalent to unit demands) throughout this paper. All the algorithms described in the following can be easily adapted to the general case. Additionally, we consider that each substrate node can host at most one virtual node, a common assumption referred to as *one-to-one case*. Consequently, the capacity of a node is necessarily either 0 or 1.

Substrate nodes (resp. edges) have a cost per unit noted $w_u \in \mathbb{N}_+$ (resp. $w_e \in \mathbb{N}_+$). The cost of a mapping is the sum of the costs of placing each virtual node and routing each virtual edge.

The Virtual Network Embedding is defined as follows.

Definition 2 (Virtual Network Embedding)**.** *The VNE problem consists in finding a minimum cost embedding of $\mathcal{G}_r$ on $\mathcal{G}_s$ that respects the capacities*

B. Related works

The Virtual Network Embedding problem is known to be NP-complete [3], even with uniform demands and topological restrictions [4], although some specific topologies can be solvable in polynomial time [5].

VNE has almost exclusively been considered as an online optimization problem, where short embedding time is required. Therefore, heuristic approaches are widely privileged and notable approaches include *two-stage* heuristic: first the node placement is decided, often via greedy heuristics ([6], [7], [8]). For example, in [8], authors propose the node Degree and Clustering Coefficient (DCC) node ranking, based on capacities and centrality measures of the substrate network, to achieve the virtual node placement. The edge routing is then computed, often with shortest-path algorithms.

Several metaheuristics such as particle swarm optimization [9], [10], ant colony [11] or genetic algorithms [12] have also been proposed. In [9], a popular one in the VNE literature is introduced as the Unified Enhanced Particle Swarm Optimization (UEPSO). On large graphs, the authors of [13] propose a Divide-and-conquer Evolutionary Algorithm (DEA) to improve an initial embedding. The virtual graph is partitioned, and mappings of subgraphs obtained from particle swarm algorithms can enhance the solution. In recent years, a focus has also been given on artificial intelligence based heuristics [14].

Currently, few works use mathematical programming to solve the VNE. The work in [15] proposes a compact formulation based on flow constraints and variables for VNE with path splitting. The linear relaxation is used to derive the rounding heuristic ViNE, used extensively in VNE literature for comparison. An adaptation of this Flow Formulation for the unsplittable case is presented in [16]. In [17], the weakness of the Flow Formulation with cycle virtual graphs is shown, in the context of finding mappings through random rounding. A new formulation is given for virtual cacti networks, which ensures that the linear relaxation can be decomposed into valid mappings. Another formulation, based on paths of the substrate network, is proposed in [18], for splittable edge routing. Since it has an exponential number of variables, it is solved through column generation and later adapted for the unsplittable case in [19], where a branch-and-price algorithm is devised.

C. Contributions

In this paper, we aim at studying the offline version of the VNE arising in the context of decision-making for resource allocation and network slice planning. We are interested in the embedding of large virtual networks, e.g. spanning entire countries. In such scenarios, the heuristics in the literature often struggle to find solutions, especially when available resources are sparse. We propose a first algorithm that can consistently provide good solutions for these instances. In Section II, we present the classical flow formulation, and then introduce a novel formulation, based on a partition of the virtual graph into smaller virtual subgraphs. Through the usage of Column Generation on this exponential formulation, we devise a Price-Branch heuristic in Section III. In Section IV, we conduct computational experiments indicating that this algorithm is often the only algorithm able to find any solution from a certain instance size. Finally, some concluding remarks are given in Section V.

II. A NEW FORMULATION BASED ON VIRTUAL GRAPH PARTITIONING

In this section we present the well-known compact flow formulation for VNE. We discuss the weakness of this formulation, leading us to introduce a formulation, the *Virtual Partition Formulation*, based on a partition of the virtual network.

A. The classical flow formulation

The compact (flow) formulation has only been introduced for the directed case in [16], but can be adapted for the undirected case. To this aim, virtual and substrate edges are given an arbitrary orientation. Hence, in the following, we consider directed copies of the networks. Then, similarly as in other flow problems [20], for each substrate edge, a symmetric arc is added to the directed version of the substrate network. The adjacent edges of a node $u \in V_s$ are denoted $\delta(u)$.

Two sets of binary variables are considered. First, the placement variable, denoted $x_{\overline{u}u}$, take the value 1 if and only if the virtual node $\overline{u} \in V_r$ is placed on substrate node $u \in V_s$. Second, the flow variable $y_{\overline{e}e_+}$ (resp. $y_{\overline{e}e_-}$) that take the value 1 if and only if the path routing virtual edge $\overline{e} \in E_r$ uses substrate directed edge $(u, v) = e \in E_s$, from u to v (resp. from v to u).

The Flow Formulation, denoted (FF), is presented in Table I where constraints (1a) ensure valid virtual nodes placement. The constraints (1b) are the flow conservation constraints and guarantee that each virtual edge is associated with a routing path in the substrate graph whose end nodes host its endpoints. Constraints (1c) are the flow departure constraints: they ensure that, when a virtual node is placed on a substrate node, adjacent virtual edges are routed through

$$
\begin{array}{llll}
\min & \sum_{u\in V_s}\sum_{\overline{u}\in V_r} w_u x_{\overline{u}u} + \sum_{e\in E_s}\sum_{\overline{e}\in E_r} w_e(y_{\overline{e}e_+}+y_{\overline{e}e_-}) & & \\
\text{s.t.} & \sum_{u\in V_s} x_{\overline{u}u} = 1 & \forall \overline{u}\in V_r & \text{(1a)}\\
 & x_{\overline{u}u} - x_{\overline{v}u} = \sum_{e\in\delta(u)} y_{\overline{e}e_+} - y_{\overline{e}e_-} & \forall \overline{e}=(\overline{u},\overline{v})\in E_r, \forall u\in V_s & \text{(1b)}\\
 & \sum_{e\in\delta(u)} y_{\overline{e}e_+} \geq x_{\overline{u}u} & \forall \overline{e}=(\overline{u},\overline{v})\in E_r, \forall u\in V_s & \text{(1c)}\\
 & \sum_{\overline{u}\in V_r} x_{\overline{u}u} \leq c_u & \forall u\in V_s & \text{(1d)}\\
 & \sum_{\overline{e}\in E_r} (y_{\overline{e}e_+}+y_{\overline{e}e_-}) \leq c_e & \forall e\in E_s & \text{(1e)}\\
 & x_{\overline{u}u}\in\{0,1\} & \forall u\in V_s, \forall\overline{u}\in V_r & \\
 & y_{\overline{e}e_+}, y_{\overline{e}e_-}\in\{0,1\} & \forall e\in E_s, \forall\overline{e}\in E_r &
\end{array}
$$

TABLE I
Flow Formulation

adjacent substrate edges. These constraints are not necessary but greatly improve the linear relaxation of the formulation. Constraints (1d) (resp. (1e)) ensure that node (resp. edge) capacities are respected. Finally, the objective function expresses an embedding cost taking into account placement and routing costs.

B. *Towards a new formulation of VNE*

As discussed in [19] and [17], the linear relaxation of the Flow Formulation can be very poor, leading to long solving time as the size of the instance increases in solver such as CPLEX. To obtain better lower bounds, the Path Formulation has been proposed in [18] and [19]. Since there is an exponential number of such variables, a Column Generation procedure is used. However, in preliminary experimentation, we have observed that the lower bound of this formulation is only slightly better than the one of the Flow Formulation, and that paths generated by the procedure are often very short, thus are not helpful to construct integer solutions.

To improve the linear relaxation, our idea is that a decomposition should consider larger subcomponents than a single virtual edge. We achieve this by a partition the virtual network into smaller subgraphs. This idea also has practical relevance: it can be expected that the virtual network is composed of smaller components, e.g. part of a slice can be dedicated to a specific service or a specific location, for which efficient subroutines can be devised.

Note that a related idea is presented in [13]. An initial embedding of a large virtual network is generated using a heuristic method. The virtual network is then partitioned into smaller subgraphs, for which individual mappings are computed heuristically. If a submapping improves upon the current best solution, it is retained. However, a major limitation of this approach is that the employed heuristics often yield poor-quality initial solutions for large-scale instances, or fail to produce any feasible solution when resource capacities are tight. To address this issue, we propose a mathematical programming framework based on a novel linear formulation.

C. *Virtual Partition Formulation (VPF)*

Consider a partition of the virtual network $\mathcal{G}_r$ into $k \geq 2$ connected subgraphs $\mathcal{H}_r^1 = (V_r^1, E_r^1), \dots, \mathcal{H}_r^k = (V_r^k, E_r^k)$, such that $V_r = \cup_{i\in\{1,\dots,k\}} V_r^i$: every virtual node is in exactly one virtual network. Some virtual edges are not contained in any subgraph: let E_r^0 denote those virtual *cut* edges. Then $E_r = \cup_{i\in\{0,\dots,k\}} E_r^i$. The subgraph that contains a node $\overline{u} \in V_r$ is denoted $\mathcal{H}_r(\overline{u})$.

The set of feasible mappings of a subgraph $\mathcal{H}_r^i$ is denoted $\mathcal{M}(i)$. The cost of a mapping $m \in \mathcal{M}(i)$ is denoted w_m. $\text{X}_{\overline{u}u}^m$ is equal to one iff $\overline{u}$ is placed on u in m, $\text{Y}_{\overline{e}e}^m$ is equal to one iff $\overline{e}$ is placed on a path that contains e in m.

Two sets of binary variables are considered. The *submapping* variables, denoted λ_m^i, take the value 1 if and only if the mapping $m \in \mathcal{M}(i)$ is selected for subgraph $\mathcal{H}_r^i$, $i \in \{1,\dots,k\}$. The flow variable $y_{\overline{e}e_+}$ (resp. $y_{\overline{e}e_-}$) that take the value 1 if and only if the path routing virtual cut edge $\overline{e} \in E_r^0$ uses substrate directed edge $(u,v) = e \in E_s$, from u to v (resp. from v to u).

The VNE can then be formulated as follows (Table II).

Constraints (2a) are the convexity constraints of the formulation, ensuring that one mapping is selected for each subgraph. Constraints (2b) (resp. (2c)) are the flow conservation (resp. flow departure) constraints for the virtual cut edges. Inequalities (2d) (resp. (2e)) ensure that substrate node (resp. edge) capacities are respected.

Note that that (VPF) is the Flow Formulation in disguised: the flow conservation and flow departure constraints remain for the virtual cut edges, whereas placement of nodes is replaced by placement of subgraph of the virtual network.

If each of the subgraphs correspond to a single virtual node, then (VPF) and (FF) are equivalent. However, for larger subgraphs, (VPF) has a greater or equal linear relaxation than (FF), and it can be strictly better in some cases.

$$
\begin{array}{llll}
\min & \displaystyle\sum_{i\in\{1,\dots,k\}}\sum_{m\in\mathcal{M}(i)} w_m\lambda_m^i + \sum_{e\in E_s}\sum_{\overline{e}\in E_r^0} w_e d_{\overline{e}} y_{\overline{e}e} & & \\
\text{s.t.} & \displaystyle\sum_{m\in\mathcal{M}(i)} \lambda_m^i = 1 & \forall i\in\{1,\dots,k\} & (\pi_i) \quad \text{(2a)}\\
& \displaystyle\sum_{\substack{m\in\mathcal{M}(\mathcal{H}_r^i),\\ s(\overline{e})\in\mathcal{H}_r^i}} \mathrm{x}_{\overline{u}u}^m\lambda_m^i - \sum_{\substack{m\in\mathcal{M}(\mathcal{H}_r^i),\\ t(\overline{e})\in\mathcal{H}_r^i}} \mathrm{x}_{\overline{v}u}^m\lambda_m^i = \sum_{e\in\delta(u)} y_{\overline{e}e_+} - y_{\overline{e}e_-} & \forall\overline{e}\in E_r^0, \forall u\in V_s & (\beta_{\overline{e}u}) \quad \text{(2b)}\\
& \displaystyle\sum_{\substack{m\in\mathcal{M}(\mathcal{H}_r^i),\\ s(\overline{e})\in\mathcal{H}_r^i}} \mathrm{x}_{\overline{u}u}^m\lambda_m^i \le \sum_{e\in\delta(u)} y_{\overline{e}e_+} & \forall\overline{e}\in E_r^0, \forall u\in V_s & (\theta_{\overline{e}u}) \quad \text{(2c)}\\
& \displaystyle\sum_{i\in\{1,\dots,k\}}\sum_{m\in\mathcal{M}(i)}\sum_{\overline{u}\in V_r^i} d_u^m\lambda_m^i \le c_u & \forall u\in V_s & (\alpha_u) \quad \text{(2d)}\\
& \displaystyle\sum_{\overline{e}\in E_r^0} y_{\overline{e}e_+} + y_{\overline{e}e_-} + \sum_{i\in\{1,\dots,k\}}\sum_{m\in\mathcal{M}(i)}\sum_{\overline{e}\in E_r^i} (\mathrm{Y}_{\overline{e}e_+}^m + \mathrm{Y}_{\overline{e}e_-}^m)\lambda_m^i \le c_e & \forall e\in E_s, & (\alpha_e) \quad \text{(2e)}\\
& \lambda_m^i\in\{0,1\} & \forall m\in\mathcal{M}(i), \forall i\in\{1,\dots,k\} & \\
& y_{\overline{e}e}\in\{0,1\} & \forall e\in E_s, \forall\overline{e}\in E_r &
\end{array}
$$

TABLE II
VIRTUAL PARTITION FORMULATION

III. A PRICE-BRANCH HEURISTIC FOR LARGE INSTANCES

As the Virtual Partition Formulation VPF contains an exponential number of variables λ, we propose a column generation based approach to solve it.

A. Column Generation for (VPF)

The *Restricted Master Problem* ($\widehat{RMP}$) of VPF is the linear relaxation of VPF limited to a small subset of λ variables. The column generation algorithm generates new columns corresponding to λ variables, through the *pricing problems*. In the present case, for each virtual subgraph $\mathcal{H}_r^i$, $i\in\{1,\dots,k\}$, the pricing problem PP_i is a VNE problem of the subgraph onto the original substrate graph. It can be solved using the Flow Formulation, with the following objective function:

$$
\begin{aligned}
v(PP_i) = \min -\pi_i + & \sum_{u\in V_r^i}\sum_{\overline{u}\in V_s}(w_u-\alpha_u)x_{\overline{u}u} \\
& + \sum_{e\in E_s}\sum_{\overline{e}\in E_r^i}(w_e-\alpha_e)y_{\overline{e}e} \\
& + \sum_{\substack{\overline{e}\in E_r^0\\ t(\overline{e})\in V_r^i}}\sum_{u\in V_s}\beta_{\overline{e}u}x_{t(\overline{e})u} \\
& - \sum_{\substack{\overline{e}\in E_r^0\\ s(\overline{e})\in V_r^i}}\sum_{u\in V_s}\beta_{\overline{e}u}x_{s(\overline{e})u} \\
& - \sum_{u\in V_s}\sum_{\substack{\overline{e}\in E_r^0,\\ s(\overline{e})\in\mathcal{H}_r^i}}\theta_{\overline{e}u}x_{s(\overline{e})u}
\end{aligned}
$$

Where the dual variables π,α,β,θ correspond to inequalities (2a)-(2d).

B. Partitioning the virtual graph

For small or large virtual networks having a known particular structure, their partitioning can be straightforward. In the general case, we propose an automatic approach based on the KaHIP algorithm [21] to obtain an interesting partition.

Given an integer k, KaHIP computes a partition into k subgraphs that minimizes the number of cut edges. Reducing the number of cut edges is desirable: it typically translates into a stronger lower bound from the column generation, and results in less y variables in the master problem. However the subgraph computed may be unconnected: the smaller parts of such subgraphs are merged with an adjacent subgraph. If the resulting k connected subgraphs are unbalanced, we transfer nodes from the largest subgraphs to smaller adjacent subgraphs. Although such transfers can raise the number of cut edges between subgraphs, this allows to obtain subgraphs of a rather equivalent size, whose pricers have equivalent solving times.

C. Heuristic pricers

Heuristic pricers are useful as solving to optimality the pricer with a solver like CPLEX is long and is only mandatory at the end of the column generation process. To devise good heuristic pricers, our idea is to consider the embedding of a virtual subgraph on a smaller part of the substrate network. To achieve this, additionally to the virtual network partition, we also propose to partition the substrate network using KaHIP to obtain several substrate subgraphs. Note that the partition does not need to be strict here: the substrate subgraphs might share some nodes, to ensure that they have the same sizes. A heuristic pricer correspond to each virtual subgraph and each substrate subgraph, hence many heuristics pricers can be devised.

An interesting strength of this approach is that it allows to control the solving time of the heuristic pricer, by choosing the size of the substrate subgraphs. Indeed, we can study, experimentally, the solving time of the Flow Formulation for a given size of a virtual network and of a substrate network.

Additionally, these heuristic pricers produces columns that does not span across the whole substrate network, which is typically observed in optimal solutions. Finally, for each virtual subgraph, it helps obtaining several columns on various parts of the substrate network. These complementary columns help the convergence of the Column Generation and the construction of a feasible integer solution.

D. Provably good solutions

To leverage the columns generated by the column generation, we propose a Price-Branch heuristic where mapping of subgraphs (i.e. λ variables) are generated until convergence is achieved, or a time limit is reached. The resulting (RMP) is then solved with integer variables through an IP solver.

The column generation is initialized with dummy columns. At early stage, we use the heuristic pricers, and when enough submappings have been generated we switch to exact pricers. After the switch the following Lagrangian bound can be computed, and gives a lower bound for the problem: $v(LB) = v(\widehat{RMP}) + \sum_{i \in \{1,\dots,k\}} v(PP_i)$.

IV. Computational experiments

In this section, we introduce our set of instances for offline VNE. Then, we conduct experiments on the Price-Branch heuristic and discuss the results.

A. Instances

In most of the existing literature, VNE is treated as an online problem. Typically, virtual networks comprising fewer than 20 nodes are generated using the GT-ITM tool [22] (see, e.g., [7]–[9], [15], [16]), with random demand values. Substrate networks are also commonly generated with GT-ITM, with random capacities assigned to nodes and links. However, the offline formulation of VNE is more suitable for certain network slicing scenarios, particularly those that involve long-term planning for the deployment of permanent slices. In such cases, although reconfigurations may occur, the initial embedding, within the network core, requires careful optimization to ensure efficiency.

We consider, for both virtual and susbtrate graphs, real core networks from the Internet Topology Zoo [23] and SND-lib [24], two widely used libraries for network problems. The virtual networks have between 50 and 85 nodes, while the substrate networks range from 150 to 200 nodes (145 nodes for Tata, 149 nodes for GTS, 197 nodes for Cogent).

For substrate networks, we consider the following tight capacities. Substrate nodes can either have a capacity of zero or one (one-to-one case), and we set 15% of the substrate nodes to have a capacity of zero. As for the edge capacities, we consider randomly generated capacities with values 2 or 3. In our preliminary tests, these capacities gave difficult but feasible instances. Finally, the node and edge costs are generated randomly between 2 and 4.

B. Results

Our Price-Branch heuristic is run with a time limit of 3600 seconds on a Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, with 90% of the time dedicated to finding columns, and 10% dedicated to finding an integer solution. The parameter k is set to $k = \lfloor |V_r|/10 \rfloor$, i.e. the virtual subgraphs have in average 10 nodes or slightly more: in our preliminary tests, as in [13], this gave the best results. For comparison, we solve the Flow Formulation using CPLEX, under the same time limit.

We also tested DCC [25], UEPSO [9] and DEA [13], the latter being the only other algorithm explicitly designed for large VNE instances. Neither DCC and UEPSO are able to find any solution, even, for UEPSO, with a runtime of 3600 seconds (which correspond to more than 10000 iterations of the algorithm). This difficulty likely stems from their reliance on shortest-path routing, which performs poorly with tight capacities. As for DEA, since it requires a feasible initial solution to operate, it also fails on all instances. These results suggest that the tested algorithms are not well-suited for such challenging scenarios. For example, DEA is primarily

Virtual Network	Substrate Network	CPLEX			Price-Branch Heuristic		
		UB	LB	gap (%)	UB	LB	gap (%)
Iris (51 nodes)	Cogent	**495**	281	**43**	535	**301**	44
	GTS	**491**	299	39	492	**323**	**34**
	Tata	**462**	280	39	474	**295**	**38**
US-Signal (61 nodes)	Cogent	780	326	58	**617**	**377**	**39**
	GTS	684	353	48	**621**	**408**	**34**
	Tata	**574**	336	41	600	**369**	**39**
Missouri (67 nodes)	Cogent	-	353	-	**612**	**408**	**33**
	GTS	737	400	48	**643**	**437**	**32**
	Tata	657	363	45	**615**	**406**	**34**
Intellifiber (73 nodes)	Cogent	-	406	-	**792**	**468**	**41**
	GTS	-	444	-	**751**	**497**	**33**
	Tata	891	414	54	**796**	**458**	**42**
OteGlobe (83 nodes)	Cogent	-	448	-	**794**	**499**	**37**
	GTS	817	504	38	**804**	**557**	**31**
	Tata	-	461	-	**769**	**515**	**39**

TABLE III
RESULT ON LARGE INSTANCES

evaluated in instances where substrate capacities are, on average, ten times higher than virtual demands [13], a very generous setting. We thus do not include these algorithms in the comparison.

The results of our Price-Branch heuristic and of CPLEX are summarized in Table III. The column UB (resp. LB) indicates the value of the best integer solution, i.e. upper bound (resp. lower bound) found by the algorithm. The gap corresponds is equal to $\frac{UB-LB}{UB}$, and is not provided if no solution is found.

They demonstrate that our algorithm successfully finds solutions for all instances, whereas CPLEX struggles on larger virtual networks. Except on the smaller virtual network Iris, our Price-Branch heuristic finds consistently better solutions than CPLEX, often by more than 10%, which indicates that the solutions of our algorithm consume less routing resources. Moreover, our method yields higher-quality lower bounds on all instances, resulting in smaller optimality gaps. However, these gaps are still important (between 30% and 40%), which indicates room for improvements in our algorithm.

V. Conclusion

In this paper, we introduced a new formulation for the Virtual Network Embedding (VNE), the Virtual Partition Formulation. To tackle large instances, we devised a Price-Branch heuristic, which provides better solutions than existing approaches for large and difficult instances.

Given the critical role of Flow Formulation efficiency in our heuristic, future research should focus on refining this formulation using polyhedral approaches, which have proven effective for other linear formulations with flow conservation constraints [20]. In particular, finding valid inequalities would be very valuable, since they could also valid for the Virtual Partition Formulation.

Another promising direction is to explore graph decomposition into overlapping parts, as demonstrated in [13]. Such a decomposition would yield stronger lower bounds, and the generated columns could be of better quality, enabling the Price-Branch heuristic to obtain better solution.

Finally, using the local search algorithm DEA [13] on the solution found by the Price-Branch heuristic could help further improve it.

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, 2018.

[2] W. da Silva Coelho, A. Benhamiche, N. Perrot, and S. Secci, "Function splitting, isolation, and placement trade-offs in network slicing," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1920–1936, 2021.

[3] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[4] A. Benhamiche, P. Fouilhoux, L. Létocart, N. Perrot, and A. Schneider, "Complexity of the virtual network embedding with uniform demands," *arXiv preprint arXiv:2501.10154*, 2025.

[5] M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, pp. 12–18, 2015.

[6] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

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

[8] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.

[9] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *International Journal of Communication Systems*, vol. 26, no. 8, pp. 1054–1073, 2013.

[10] P. Zhang, Y. Hong, X. Pang, and C. Jiang, "Vne-hpso: Virtual network embedding algorithm based on hybrid particle swarm optimization," *IEEE Access*, vol. 8, pp. 213 389–213 400, 2020.

[11] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic," in *2011 IEEE international conference on communications (ICC)*. IEEE, 2011, pp. 1–6.

[12] P. Zhang, H. Yao, M. Li, and Y. Liu, "Virtual network embedding based on modified genetic algorithm," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 481–492, 2019.

[13] A. Song, W.-N. Chen, Y.-J. Gong, X. Luo, and J. Zhang, "A divide-and-conquer evolutionary algorithm for large-scale virtual network embedding," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 566–580, 2019.

[14] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[15] 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, 2011.

[16] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 356–368, 2013.

[17] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2071–2084, 2019.

[18] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 334–348, 2015.

[19] L. F. Moura, L. P. Gaspary, and L. S. Buriol, "A branch-and-price algorithm for the single-path virtual network embedding problem," *Networks*, vol. 71, no. 3, pp. 188–208, 2018.

[20] C. Raack, A. M. Koster, S. Orlowski, and R. Wessäly, "On cut-based inequalities for capacitated network design polyhedra," *Networks*, vol. 57, no. 2, pp. 141–156, 2011.

[21] P. Sanders and C. Schulz, "Think locally, act globally: Highly balanced graph partitioning," in *International Symposium on Experimental Algorithms*. Springer, 2013, pp. 164–175.

[22] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE INFOCOM'96. Conference on Computer Communications*, vol. 2. IEEE, 1996, pp. 594–602.

[23] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[24] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[25] F. Zhu and H. Wang, "A modified aco algorithm for virtual network embedding based on graph decomposition," *Computer Communications*, vol. 80, pp. 1–15, 2016.